# *PRE-PRINT*
# APPENDIX

## Supplementary Information

for the paper

# *A robot that walked 65 km on a single charge: energy-effective and reliable locomotion using trajectory optimization and stabilization from reflexes*

For submission to International Journal of Robotics Research

Most recent modification on April 29, 2012.

| | |
|---|---|
| Pranav A. Bhounsule, | Disney Research, Pittsburg, PA (formerly Cornell), |
| Jason Cortell, | Mechanical Engineering, Cornell University, |
| Bram Hendriksen, | Delft University of Technology, The Netherlands, |
| J.G. Daniël Karssen, | Delft University of Technology, The Netherlands, |
| Chandana Paul, | Mechanical Engineering, Cornell; Applied Sciences, Harvard, |
| Andy Ruina, | Corresponding author: ruina@cornell.edu<br>Mechanical Engineering, Cornell University, Ithaca NY |

This appendix provides some details about the robot Ranger, its hardware, software, simulation, optimization and the design of its controller. Please read the main paper before looking at this appendix. Even more details, including photos, videos and $100^+$ reports are on Cornell's www site:

http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/ranger/Ranger2011/

## Contents

### The Machine

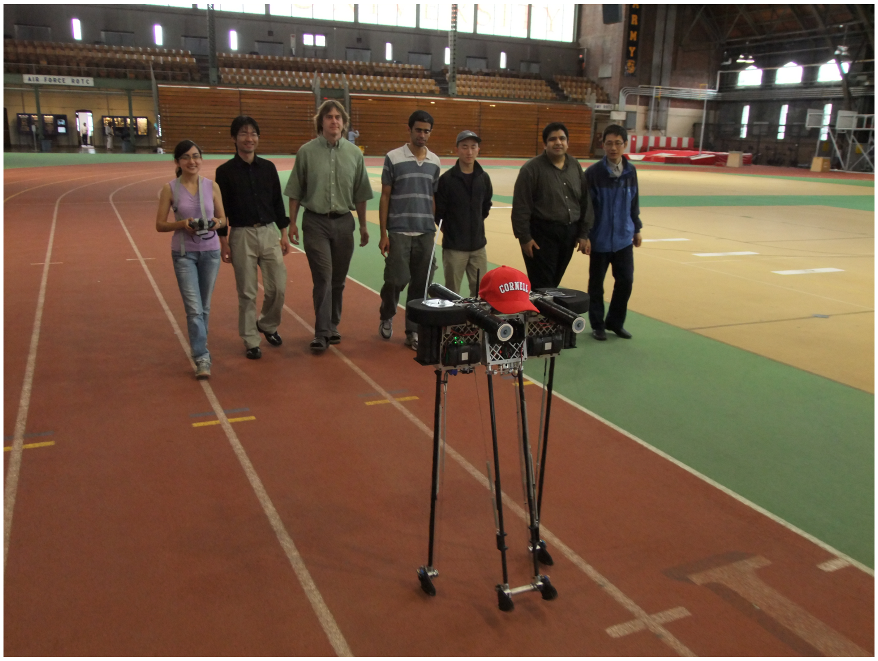
1. Introduction (and notation) .......... 2
2. Mechanical hardware .......... 6
3. Electronics .......... 14
4. Software on robot microcontrollers .......... 27

### Simulation and Control

5. Equations of motion .......... 41
6. Benchmark tests of the equations of motion .......... 48
7. System identification for mechanical parameters .......... 52
8. System identification for motors and gearboxes .......... 57
9. Smoothings for simulations and optimizations .......... 64
10. Benchmark tests for optimal trajectory calculations .......... 66
11. Trajectory optimization: fine-grid .......... 74
12. Trajectory optimization: coarse-grid .......... 78
13. Finite state machine for control of walking .......... 81

# 1 Introduction (and notation)

Ranger is a 4-legged bipedal robot built to be reliable while using little energy. Its four side-by-side legs are meant to mimic 2 legs. The outer pair of legs move together, acting as one leg, and so do the inner pair, hence the oxymoronic description '4-legged biped'. This robot lives, essentially, in two spatial dimensions (the sagittal plane). It has 3 main motors and 3 main internal degrees of freedom (hip and two ankles). Its culminating achievements are a 65 km walk and, later, the ability to walk with a total cost of transport of 0.19. Both of these seem to be bests for legged robots. This appendix describes some of the details of Ranger's construction and control.



| | |
|---|---|
| Total steps | 186,076 |
| Total time | 110,942 s (= 30 hrs 49 min 2 sec) |
| Number of laps | 307.75 |
| Lap distance | 212 m (= 0.132 miles) |
| Total distance | 65,243 m (= 65.24 km = 40.54 mi) |
| Average time per step | 0.6 s |
| Average distance per step | 0.35 m (= 13.78 in) |
| Average speed | 0.59 m/s (= 2.12 km/h = 1.32 mph) |
| Total power | 16 W |
| Power used by motors | 11.3 W |
| Power used by computers and sensors | 4.7 W |
| Total energy used | 493 watt-hours |
| Battery | 25.9 V Li-ion |
| Total robot mass | 9.91 kg (= 21.85 lb) |
| Battery mass | 2.8 kg (= 6.3 lb) |
| Total cost of transport (TCOT) | 0.28 (later lowered to 0.19) |

**Ranger's ultra-marathon walk.** On 1-2 May 2011 [9], Ranger walked non-stop for 40.5 miles (65 km) on Cornell's Barton Hall track without recharging or being touched by a human. Some of the crew that worked on Ranger are shown walking behind Ranger during the 65 km walk. Basic data are in the table above.

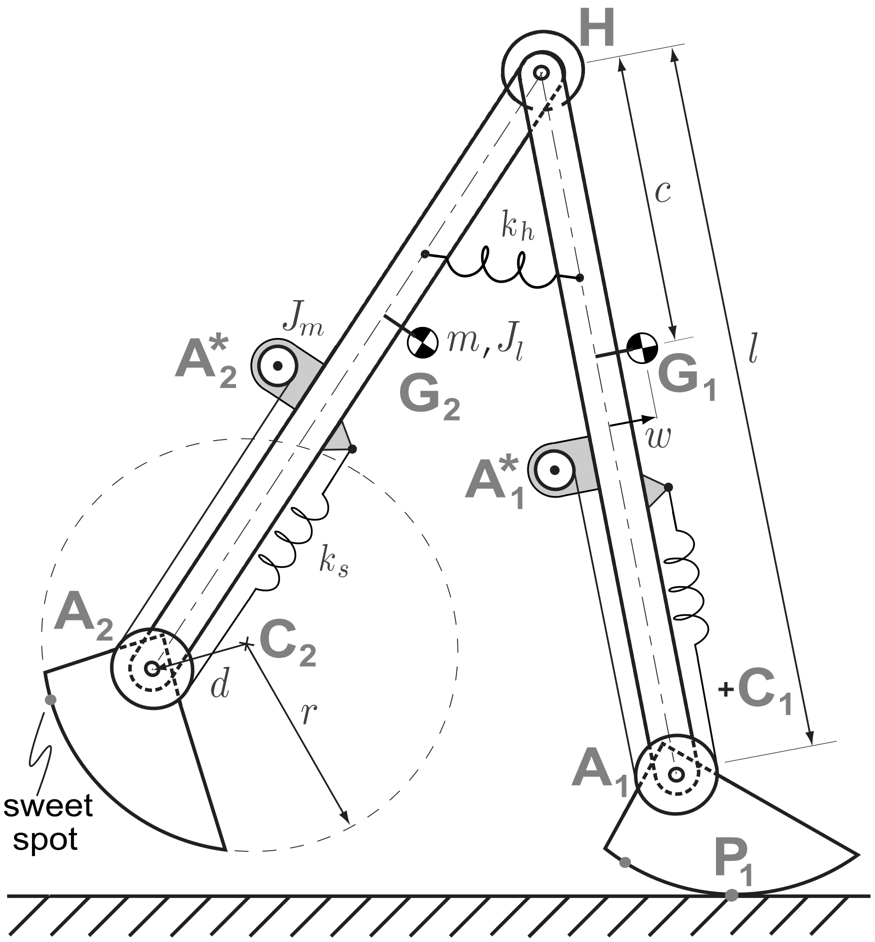


Figure 1: **Robot dimensions and feet geometry.** The feet bottoms are roughly circular arcs with radius $r$. The ankle joints $A_1$ and $A_2$ are offset from the center of circle by the distance $d$. As dictated by the geometry of circles the contact points $P_1$ and $P_2$ are always directly below the center of the circles $C_1$ and $C_2$, respectively, in level-ground walking. There is one foot configuration in which the ankle joint lies on the line joining the center of the circle and the contact point. For vertical ground forces this is a natural equilibrium position for the feet; it takes no ankle torque to hold the foot in this position. The contact point is then that part of the foot circular arc that is closest to the ankle. We call this point on the foot the 'sweet-spot'. The ankle motors are connected to the ankle joints via cables that we approximate as linear springs. The ankle motors ($A_1^*$, $A_2^*$) are actually nearly coincident with the hip H, but are separated in this diagram for clarity.

## 1.1 Notation

Variables used in this appendix are listed on the next two pages, some with respect to the side-view robot schematic figure 1 below. The main internal degrees of freedom are the hinges at the ankles ($A_1$, $A_2$) and hip (H). Because the ankle drive cables are elastic, the motors at $A_1^*$ and $A_2^*$ add two additional internal degrees of freedom.

### 1.1.1 Robot parameters

| Symbol | Value | Parameter description |
|---|---|---|
| $\ell$ | 0.96 m | Leg length. |
| $r$ | 0.2 m | Foot radius. |
| $d$ | 0.11 m | Ankle eccentricity. |
| $w$ | 0 | Fore-aft distance of COM. |
| $c$ | 0.15 m | Distance of COM from hip along the leg. |
| $k_h$ | 7.6 N m/rad | Hip Spring constant. |
| $k_s$ | 14 N m/rad | Ankle Spring constant. |
| $J_\ell$ | 0.45 $\mathrm{kg\,m^2}$ | Inertia of legs about COM. |
| $J_{hip}$ | 0.55 $\mathrm{kg\,m^2}$ | Inertia of legs about hip hinge. |
| $m$ | 4.96 kg | Mass of a leg. |
| $M_{tot}$ | 9.91 kg | Total robot mass. |
| $g$ | 9.81$\mathrm{m/s^2}$ | Gravitational constant. |
| $\gamma$ | 0 | Ground slope (positive value is downhill). |
| $C_{1FW}$ | 0.05 N s/m | Coefficient of viscous friction between ground and stance leg. |
| $C_{2FW}$ | 0.05 N s/m | Coefficient of viscous friction between ground and trailing leg... in double stance. |

### 1.1.2 Motor parameters

| Symbol | Value | Parameter description |
|---|---|---|
| $G_H$ | 66 | Hip gear ratio. |
| $G_A$ | 34 | Ankle gear ratio. |
| $K$ | 0.018 N m/A | Motor torque constant. |
| $R$ | 1.3 Ω | Motor terminal resistance. |
| $V_c$ | 0.7 V | Contact voltage of the brush-commutator interface. |
| $J_m$ | 0.002 $\mathrm{kg\,m^2}$ | Motor inertia. |
| $\mu_H$ | 0.1 | Coefficient of current dependent constant friction: hip motor. |
| $\mu_A$ | 0.1 | Coefficient of current dependent constant friction: ankle motor. |
| $C_{H1}$ | 0.01 N s/m | Coefficient of viscous friction: hip motor. |
| $C_{H0}$ | 0.1 N | Coefficient of constant friction: hip motor. |
| $C_{A1}$ | 0.01 N s/m | Coefficient of viscous friction: ankle motor. |
| $C_{A0}$ | 0.1 N | Coefficient of constant friction: ankle motor. |
| $P_{\text{fixed}}$ | 5.15 W | Power used by microprocessors, sensors and motor controller. |

### 1.1.3 Other variables

| Symbol | Variable description |
|---|---|
| $t$ | time. |
| $q_1$, $r_1$ | absolute angle made of stance foot wrt. vertical after and before heelstrike. |
| $q_2$, $r_2$ | relative angle between stance foot and stance leg after and before heelstrike. |
| $q_3$, $r_3$ | relative angle between legs; also hip angle after and before heelstrike. |
| $q_4$, $r_4$ | relative angle between swing foot and swing leg after and before heelstrike. |
| $q_{2m}$, $q_{4m}$ | motor angles at points $A_1^\star$ and $A_2^\star$ after heel-strike respectively. |
| $r_{2m}$, $r_{4m}$ | motor angles at points $A_1^\star$ and $A_2^\star$ before heel-strike respectively. |

| | |
|---|---|
| $x$, $y$ | world reference frame, $x$ in walking direction and $y$ is against gravity. |
| $x_h$, $y_h$ | $x$ and $y$ co-ordinate of the hip joint respectively. |
| $I_i$ | Motor current. $i = 2, 3, 4$ at the points $A_1^\star$, $H$ and $A_2^\star$ respectively. |
| $d_{\textbf{step}}$ | Step length. |
| $t_{\textbf{step}}$ | Step time. |
| $v_{\textbf{step}}$ | Step velocity. |
| $M$ | Mass at the hip (for benchmarks in appendices 6 and 10). |
| $E$ | Energy. |
| $P$ | Power. |
| $N_i$ | Number of grid points, $i = ss$ (single stance) or $i = ds$ (double stance). |
| $\chi$ | Robot state vector and includes angles and angular rates. |
| $T_{1FW}$ | Torque between ground and stance foot. ($T_{1FW} = -C_{1FW}\dot{q}_1$). |
| $T_{2FW}$ | Torque between ground and trailing stance foot in double stance... ($T_{2FW} = -C_{2FW}(\dot{q}_1 + \dot{q}_2 - \dot{q}_3 - \dot{q}_4)$). |
| $T_3$ | Hip Motor Output Torque ($= G_H K I_3 - T_{fH}(I_3, \dot{q}_3)$) |
| $T_{fH}(I_3, \dot{q}_3)$ | Hip Motor Friction Torque ($= \mu_H \mathrm{sgn}(\dot{q}_3) G_H K \lvert I_3 \rvert + C_{H1}\dot{q}_3 + C_{H0}\mathrm{sgn}(\dot{q}_3)$). |
| $T_{fA}(I_i, \dot{q}_{im})$ | Ankle Motor Friction Torque ... ($= \mu_A \mathrm{sgn}(\dot{q}_{im}) G_A K \lvert I_i \rvert + C_{A1}\dot{q}_{im} + C_{A0}\mathrm{sgn}(\dot{q}_{im})$) where $i = 2, 4$. |
| $T_{2S}$ | Ankle Spring Torque ($T_{2S} = k_s(q_{2m} - q_2)$). |
| $T_{3S}$ | Hip Spring Torque ($T_{3S} = k_h q_3$). |
| $T_{4S}$ | Ankle Spring Torque ($T_{4S} = k_s(q_{4m} - q_4)$). |
| $F_{2S}$, $F_{2S}^{'}$ | Tensional force in the ankle cables at joint associated with dof. $q_2$. |
| $F_{4S}$, $F_{4S}^{'}$ | Tensional force in the ankle cables at joint associated with dof. $q_4$. |
| $H_i$, $V_i$ | Horizontal and vertical reaction forces respectively at joint $i$. |
| $H_i^\star$, $V_i^\star$ | Horizontal and vertical impulse respectively at joint $i$. |
| $\overrightarrow{g}$ | gravity vector ($= g\sin(\gamma)\hat{\imath} - g\cos(\gamma)\hat{\jmath}$). |
| $\overrightarrow{\omega}_1$ | Absolute angular velocity of $G_1$ after heelstrike ($\omega_1 = \dot{q}_1 + \dot{q}_2$). |
| $\overrightarrow{\omega}_2$ | Absolute angular velocity of $G_2$ after heelstrike ($\omega_2 = \dot{q}_1 + \dot{q}_2 - \dot{q}_3$). |
| $\overrightarrow{\omega'}_1$ | Absolute angular velocity of $G_1$ before heelstrike ($\omega_1' = \dot{r}_1 + \dot{r}_2$). |
| $\overrightarrow{\omega'}_2$ | Absolute angular velocity of $G_2$ before heelstrike ($\omega_2' = \dot{r}_1 + \dot{r}_2 - \dot{r}_3$). |
| $\overrightarrow{\alpha}_1$ | Absolute angular acceleration of $G_1$, the stance leg ($\alpha_1 = \ddot{q}_1 + \ddot{q}_2$). |
| $\overrightarrow{\alpha}_2$ | Absolute angular acceleration of $G_2$, the swing leg ($\alpha_2 = \ddot{q}_1 + \ddot{q}_2 - \ddot{q}_3$). |
| $\overrightarrow{v}_H$ | Velocity of point $H$ after heelstrike, ($= \dot{x}_h\hat{\imath} + \dot{y}_h\hat{\jmath}$). |
| $\overrightarrow{v}_{H'}$ | Velocity of point $H$ before heelstrike, ($= \dot{x}_{h'}\hat{\imath} + \dot{y}_{h'}\hat{\jmath}$). |
| $\overrightarrow{a}_H$ | Acceleration of point $H$ ($= \ddot{x}_h\hat{\imath} + \ddot{y}_h\hat{\jmath}$). |
| $\overrightarrow{v}_{G_1}$ | Velocity of point $G_1$ after heelstrike ($= \overrightarrow{v}_H + \overrightarrow{\omega}_1 \times \overrightarrow{r}_{G_1/H}$). |
| $\overrightarrow{v}_{G_1'}$ | Velocity of point $G_1$ before heelstrike ($= \overrightarrow{v}_{H'} + \overrightarrow{\omega}_1' \times \overrightarrow{r}_{G_1'/H'}$). |
| $\overrightarrow{a}_{G_1}$ | Acceleration of point $G_1$ ($= \overrightarrow{a}_H - \omega_1^2\ \overrightarrow{r}_{G_1/H} + \overrightarrow{\alpha}_1 \times \overrightarrow{r}_{G_1/H}$). |
| $\overrightarrow{a}_{G_2}$ | Acceleration of point $G_2$ ($= \overrightarrow{a}_H - \omega_2^2\ \overrightarrow{r}_{G_2/H} + \overrightarrow{\alpha}_2 \times \overrightarrow{r}_{G_2/H}$). |
| $\overrightarrow{v}_{G_2}$ | Velocity of point $G_2$ after heelstrike ($= \overrightarrow{v}_H + \overrightarrow{\omega}_2 \times \overrightarrow{r}_{G_2/H}$). |
| $\overrightarrow{v}_{G_2'}$ | Velocity of point $G_2$ before heelstrike ($= \overrightarrow{v}_{H'} + \overrightarrow{\omega}_2' \times \overrightarrow{r}_{G_2'/H'}$). |
| $\overrightarrow{r}_{H/P_1}$ | Position vector from point $P_1$ to point $H$ and so on. |

| | |
|---|---|
| $\vec{P}$ | Force on the trailing foot from ground in double stance ($= P_x\hat{\imath} + P_y\hat{\jmath}$). |
| $\vec{P}^\star$ | Impulse on the trailing foot from ground in double stance ($= P_x^\star\hat{\imath} + P_y^\star\hat{\jmath}$). |
| $\vec{H}_{/P_1}$ | Angular momentum about point $P_1$ and so on. |
| $\dot{\vec{H}}_{/P_1}$ | Rate of change of angular momentum about point $P_1$ and so on. |
| $\vec{M}_{/P_1}$ | External angular moment about point $P_1$ and so on. |
| $\lvert . \rvert$ | Absolute value function ($\lvert x \rvert = x$ for $x > 0$ and $\lvert x \rvert < -x$ for $x \leq 0$). |
| $[.]^+$ | Ramp function ($[x]^+ = x$ for $x > 0$ and $[x]^+ = 0$ for $x \leq 0$). |
| **sgn**(.) | Signum function ($\text{sgn}(x) = x/\lvert x \rvert$). |

## 2 Mechanical hardware

Figure 2 shows the actual arrangement of Ranger's components. The two inner feet are linked directly by a hollow shaft, while the outer feet are driven by a single motor through separate cables and are linked by the truss above the hip hinge. Consistent with the facetious 'biped' moniker one can think of the outer leg (singular) and the inner leg. The steering is a fourth controlled degree of freedom: an electric motor twists the inner feet back and forth with each step, allowing Ranger to make gradual turns. The steering has little effect on the two-dimensional (sagittal plane) walking gait. The ankle joint motor has two main functions: pushing off at the end of stance and flipping up so the feet clear the ground during swing.

Most of the robot mass is near the hip. With light legs and light feet, leg swing can be quick using small torque. Further, the stance leg dynamics become nearly decoupled from the swing leg dynamics. Ranger's legs are hollow carbon fiber tubes and the feet are cut from high-strength 7075 aluminum alloy stock. The ankle joint motors are located on the hip axis.

Structurally, Ranger has three 'thighs': one aluminum box supporting the inner pair of legs, and one more for each of the outer two legs. The outer leg boxes are spanned by an aluminum top truss, which also supports electrical wires and ankle-drive cables. Protection against the occasional fall is provided by protruding foam 'eye' stalks and foam 'ears'.

The robot has a total mass of 9.91 kg and a leg length of 1 m, measured from the bottom of the foot up to the hip axis. The total height is about 1.1 meters, not including antennae.

### 2.1 Hip drive

Hip actuation is via a 46-watt-nominal brushed direct current motor (Faulhaber #2657W012CR) and a 66:1 planetary gearhead. The hip motor and gearhead are mounted in the outer leg box, aligned axially with the hip joint. The gearhead output shaft is attached to the inner leg via a flexible coupling, to prevent bearing loads due to slight construction misalignment between the hip bearing mounts and the motor mount.

### 2.2 Ankle drive system

Each pair of feet is actuated by a 46-watt-nominal brushed direct current motor (Faulhaber #2657W012CR) with a 43:1 planetary gearhead.

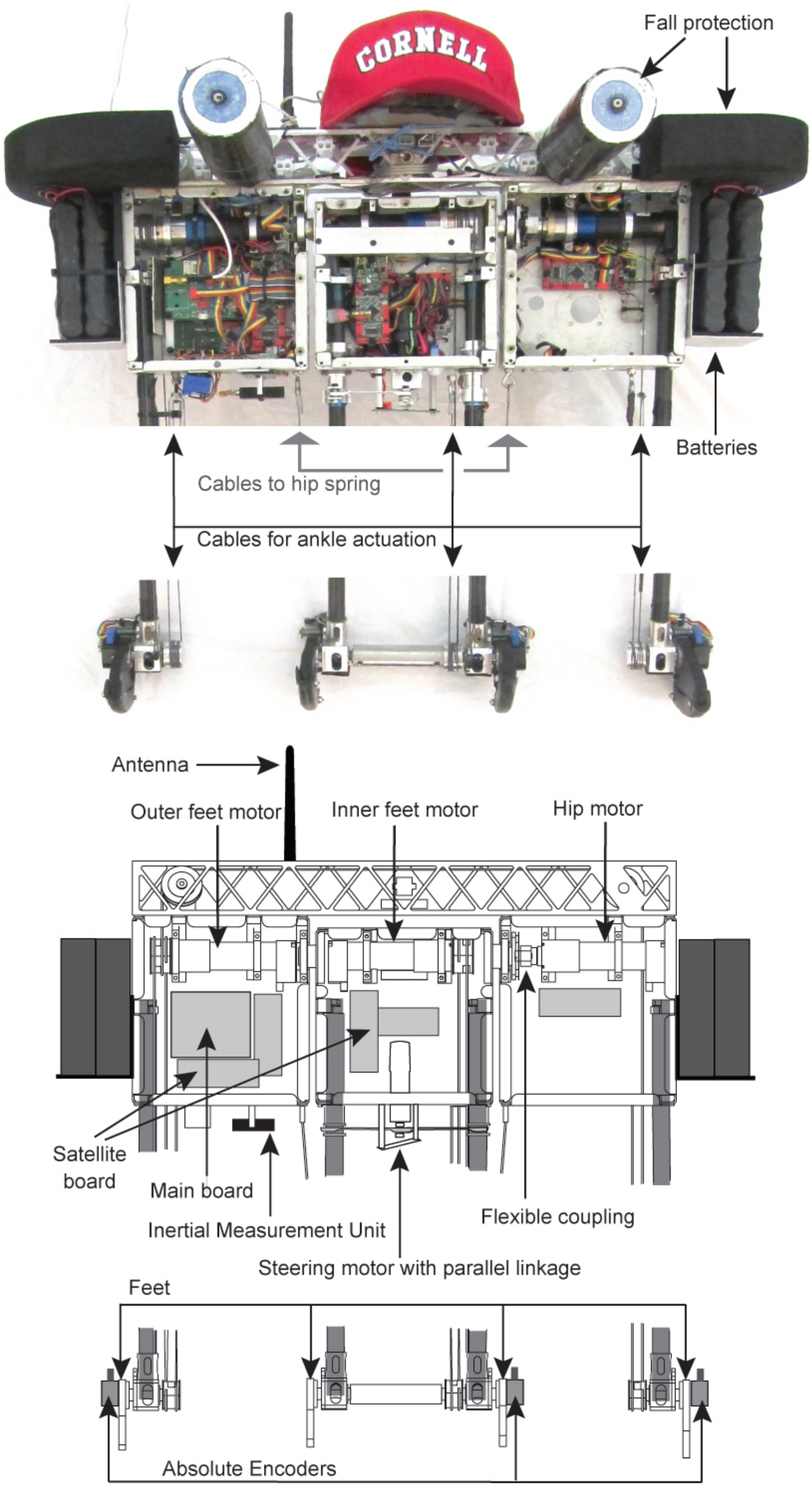

Fall protection
CORNELL
Batteries
Cables to hip spring
Cables for ankle actuation
Antenna
Outer feet motor
Inner feet motor
Hip motor
Satellite board
Main board
Inertial Measurement Unit
Flexible coupling
Steering motor with parallel linkage
Feet
Absolute Encoders

Figure 2: **Overall layout, front view.** In the photograph the boxes are open. The foam assemblies for fall protection ('eyes' and 'ears') serve no other purpose. The hat is only decorative. The main and satellite boards contain most of the electronics. More details are in the text and following illustrations.

**Cables and springs.** The motors drive the feet through a system of thin stainless steel cables (figure 3). The cable at the back of the ankle joint (analogous to the Achilles tendon in humans) goes directly from the foot pulley to the motor pulley. However, there is no direct connection between the front of the foot pulley and the front of the motor pulley. Instead, the two cable ends coming off the fronts of the pulleys are attached to a pair of springs, which are in turn attached to fixed points on the leg assembly. The spring at the foot pulley causes the foot to flip upward when the motor relaxes the (Achilles tendon) drive cable. The spring at the motor pulley balances the force of the ankle spring, so the motor need not waste power continually fighting it. It also prevents slack when the motor is off. The pre-stressed springs, chosen for a small spring constant and a large initial extension, behave almost like constant force springs, allowing the motor to move the ankle through its full range of motion with minimal interference from the springs.

The inner feet are rigidly connected so that they can be actuated by one cable. For the outer feet two cables are used, wound in tandem on a single motor pulley (see figure 3).

The ankle-drive cables need to be small diameter to allow long life when used over small-diameter pulleys. We used 7 x 19 strand core low-stretch nylon-coated stainless steel cable made by Sava Industries. The cable for the inner feet (model 2050SN2) had an overall (with coating) diameter of 1.59 mm, an uncoated (stainless-only) diameter of 1.19 mm, and a breaking strength of 1200 N. The outer feet used model 2037SN cable, with a coated diameter of 1.17 mm, an uncoated diameter of 0.96 mm, and a breaking strength of 710 N. These cables had non-negligible compliance, stretching under tension and thus, with the motors, making up series-elastic actuators. Two lengths of cable are used to drive the outer ankle pair, but only one for the inner ankle pair (see figure 3). To ensure symmetry of control and behavior for both pairs of ankles, we needed to appropriately match all three cable compliances as follows:

**Matching cable compliances.** For a laterally symmetric gait the cables between the outer ankle motor and the ankles are in parallel (e.g. for a small rotation of the outer ankle motor, with the ankles joints held fixed, each cable stretches an equal distance). Thus the outer cable's spring constants add. For gait symmetry the single cable driving the inner ankles should have the same spring constant as the combined value from the outer ankle cables, so the inner cable should be twice as stiff as each of the outer-ankle cables. To double the stiffness, a larger cable with double the cross-sectional area was used for the inner feet. Further, the cable to the left outer ankle needs to go up and across the top truss before heading down to the ankle, making the left cable about twice as long as the right outer ankle cable, and giving it twice the compliance. To make the left and right Achilles tendons match, the left cable was threaded through a length of thin carbon fiber tubing running almost the full length of the leg and glued securely, thus reducing the stretchable portion of the cable length.

**Attachments.** Stress concentrations at the connections of the tendon cables with the carbon fiber tubes were reduced by tapering the tube thickness at those points, and by leaving the nylon cable coating in place for several centimeters into the tube. Farther inside the tube, the nylon was stripped off the cable, allowing a strong epoxy bond to the stainless steel strands within. In an early incarnation steel was used instead of carbon-fiber and the connection with the cable was with solder. One robot failure was from tendon cable breakage due to corrosion from acidic flux residues from this soldering.

The cable ends are clamped inside the pulleys.

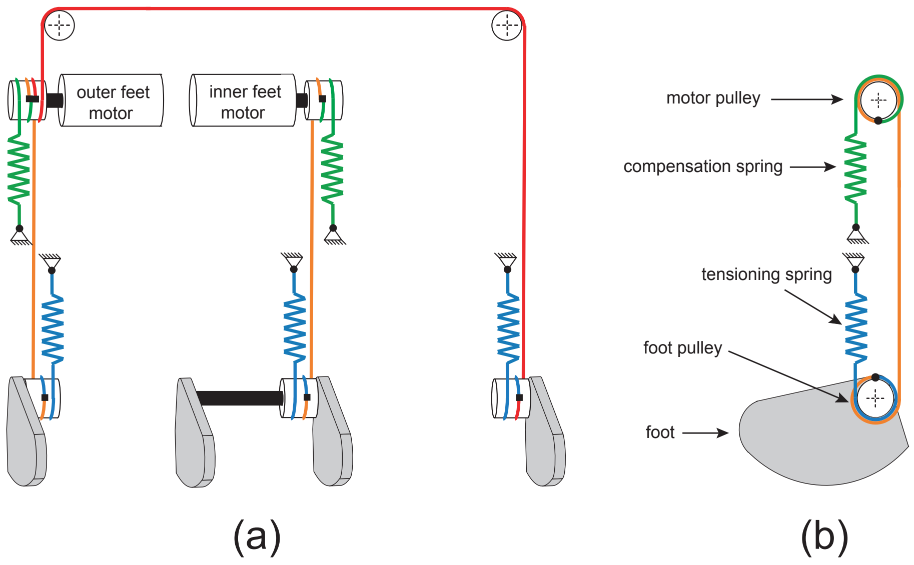


Figure 3: **Ankle actuation.** **(a)** Perspective schematic, **(b)** Side schematic of one leg. One motor controls the inner ankles and one controls the outer ankles. For each leg pair ankle extension (e.g. push-off) is powered by a single cable and retraction (for foot ground clearance) is powered by a return spring. Another return spring on the motor prevents cable slack when the motor is off, and also balances out the force of the ankle return spring so that the motor doesn't have to.

Because there are angle sensors at both ends of the drive cables (at the ankle joints and also at the drive motor shaft), the ankle torque can be measured by multiplying the cable stretch by the cable stiffness. In the end, however, we found no need for this torque measurement in the controller.

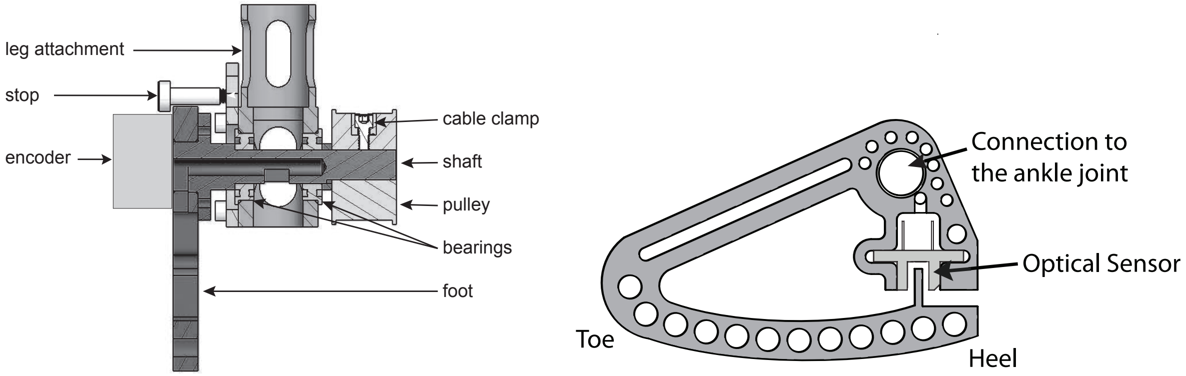
### 2.3 Ankle joints

There is some subtlety in the ankle design (figure 4(a)). The ankle needs to be light and strong while also providing:

1. a secure connection to the carbon fiber leg tube,
2. a low-friction rotational joint with minimal play,
3. a stop to limit the motion of the foot,
4. and a path to take the foot contact sensor cable out of the foot and up the leg tube.

while holding up to hundreds of thousands of heel-strike impacts.

The 7075 aluminum ankle shaft is partially hollow and has an opening in the middle so that the sensor cable can go from the foot up into the hollow leg tube. The sensor cable is wrapped around the foot shaft once or twice, spiraling outward. Thus, when the shaft rotates, the resulting bend in the cable is distributed over a substantial length, minimizing local strains and maximizing



(a) Cross-section of the ankle

(b) Foot side view

Figure 4: **Ankle and foot. a)** Front view of a cross-section of the foot-ankle assembly, **b)** Side view of the foot. The top of the foot hits the stop when the top of the foot is approximately vertical. The 'stop' includes a spring-loaded electrical contact, thus acting as both a mechanical stop and an electrical limit switch.

flexure life. The cable used was selected for long life in high-flex environments, with extra-fine-gauge conductors made of high-strength copper alloys. (e.g., 3M high-flex ribbon cable, series 3319.)

Each ankle shaft is supported by two ball bearings, flanged for ease of fabrication. The foot screws to a flange machined in one piece with the shaft. The opposite end of the ankle shaft has a flat machined into it, forming a "D" in cross section. The foot pulley clamps on with a matching D-shaped opening, thus preventing slippage and play. Initial student designs used just a set screw or just a clamp to connect the pulley to the foot-drive shaft. Such arrangements were found to be unreliable with too much slip or play or both.

## 2.4 Feet

The foot shape is shown in figure 4(b). Its bottom edge is a portion of a larger circular arc centered above and in front of the ankle joint (point $C_2$ in figure 1 on page 3). The foot is shaped and positioned such that when made to stand in its equilibrium position on flat ground the point of foot contact (the sweet spot) is part of the circular arc. This point is near the heel. It takes zero torque to maintain the foot in this position when the ground reaction is vertical.

The feet are also load cells cut in an open C shape. A vertical load on the bottom of the foot closes the gap at the heel. The size of this gap is measured with an optical sensor. Finite element analysis was used to design the foot so that expected loads would cause deformation within the range of the optical sensor. Further, the gap at the back was sized to close for large loads, thus preventing plastic deformation of the foot from unusually large heel-strike forces.

## 2.5 Support structure (hips/thighs)

The hips/thigh for the outer legs are constructed of two folded and spot-welded 5052 aluminum sheet boxes, with a top truss (machined from 6061-T6 aluminum square tubing) connecting the two boxes. The inner leg is constructed of one box (figure 5). Fully closed box structures were chosen for their light weight, high stiffness, high strength, and ease of fabrication. The motor mounts and

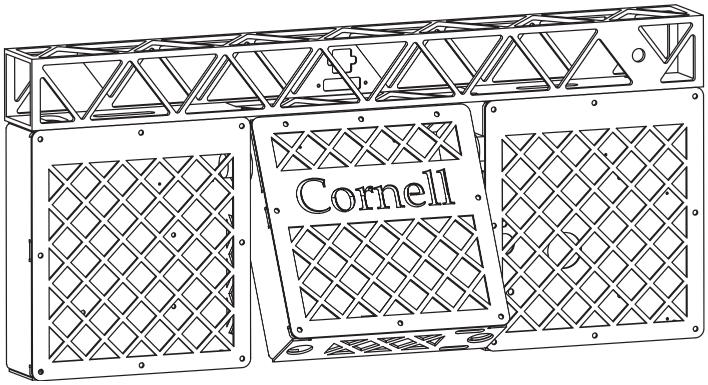


Figure 5: **Legs supporting structures (hips/thighs).** Each outer leg has one closed aluminum box made from folded and welded sheet stock. The pair of inner legs has a similar closed box. The outer boxes are joined by a truss made from from a hollow aluminum box beam with triangles machined out to reduce weight. The inner box is hinged to the outer boxes by the hip axle which runs parallel to, and below, the truss.

top truss were bonded with Mereco Metregrip 303 epoxy adhesive, with screws as a backup. The rectangular boxes were also convenient for enclosing electronics, motors, batteries, etc. Because the box stiffness and strength is reduced by orders of magnitude when the cover plates are removed, the robot cannot operate without the box covers.

For stationary bench testing of the hip and outer ankle motors, the top bar provides sufficient support to resist motor torques even with the covers off, but the inner box does not have such support. For testing with the inner box cover removed an angle bracket was fastened across the motor mounts, to provide some supplemental support.

## 2.6 Hip spring

The robot has a pair of tension springs connecting the inner legs with the outer legs. One end of each spring is connected to the inner legs, near the bottom, while the opposite end is connected to the outer boxes. The spring length is minimum when the legs are parallel; therefore, because they are pre-stretched, they effectively act as a torsion spring at the hip joint, pulling the legs towards a parallel configuration (see figure 6b). Using figure 6, we can calculate the torque at the hip as

$$\begin{aligned} T_{\text{hip}} &= 2kab\left(1-\frac{\ell_0}{\ell}\right)\sin(\theta) \\ &= 2kab\left(1-\frac{\ell_0}{\sqrt{a^2+b^2-2ab\cos(\theta)}}\right)\sin(\theta) && (1) \\ &\approx \underbrace{2kab\left(1-\frac{\ell_0}{b-a}\right)}_{k_h}\theta \equiv k_h\theta && (2) \end{aligned}$$

where $T_{\text{hip}}$ is hip spring torque, $k$ is the spring constant of the tension spring, $a$ and $b$ are distances from the hinge joint to the connection point on the inner and outer leg respectively, $\ell_0$ and $\ell$ are the lengths of the spring when the legs are parallel and when the hip is at an angle $\theta$, respectively. $k_h$

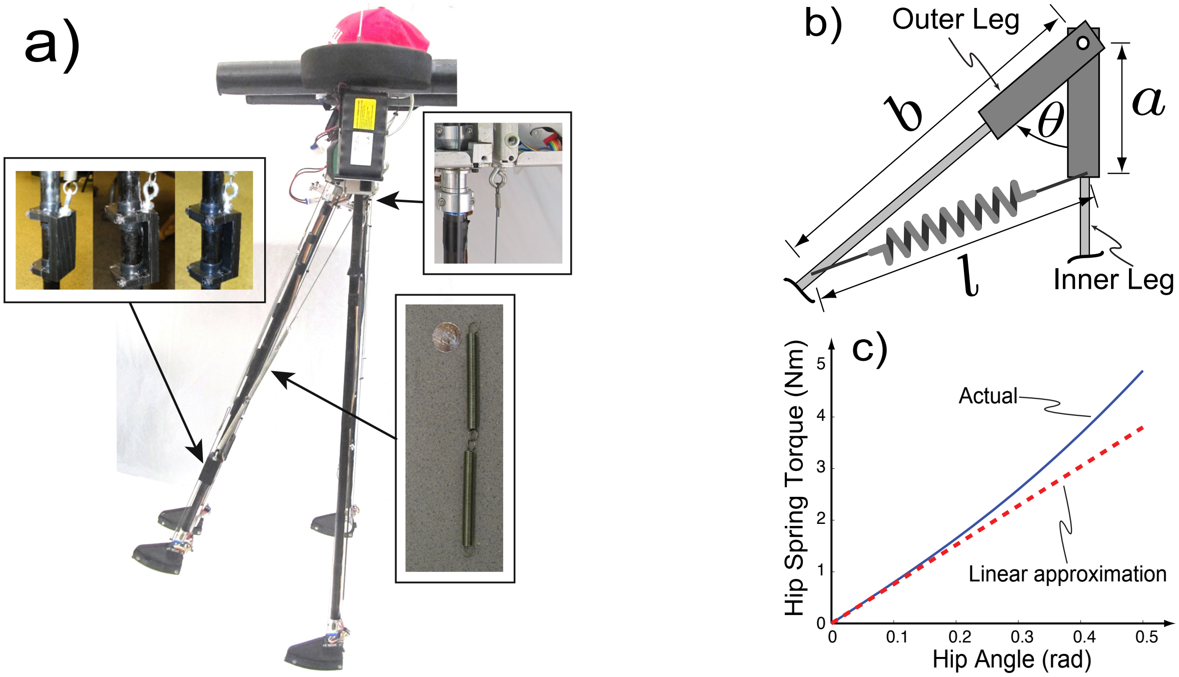
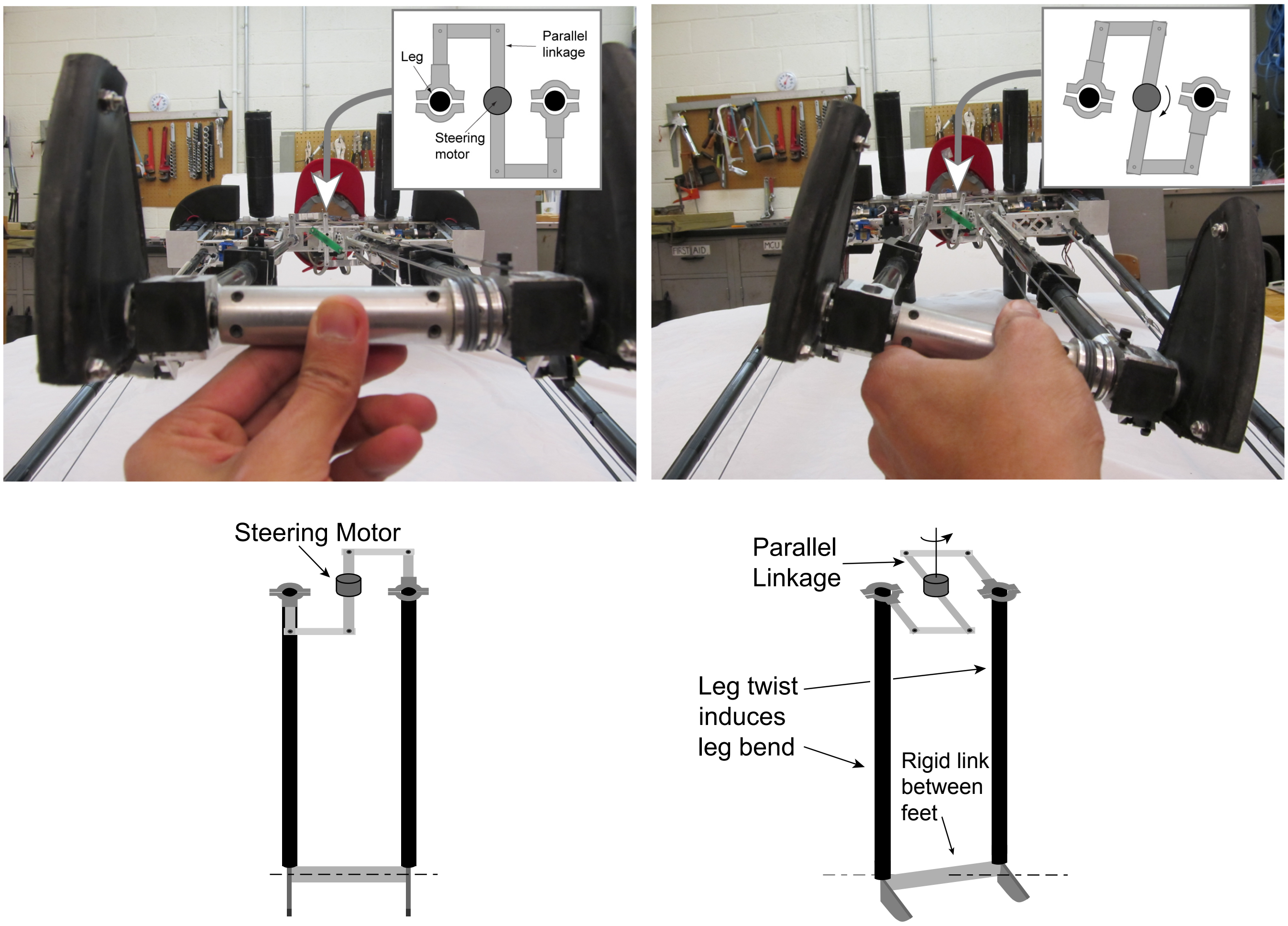


Figure 6: **Hip Spring.** The hip spring acts as a torsional spring tending to keep the outer legs parallel with the inner legs. Its purpose is to speed the leg swing with less motor effort. **a)** a side-view photograph showing also the spring and attachments. **b)** Schematic diagram of the hip spring, **c)** Plot of hip spring torque vs hip angle. Blue solid line is hip spring torque calculated using equation 1. The linear approximation is equation 2, which we use in simulations. The nearly linear behavior follows from the spring pre-stretch, which gives nearly constant tension.

is the small-angle effective spring constant. The spring stiffness $k$ was chosen so that the hip motor barely had the strength (6 N m) to hold its leg at an angle of 0.4 rad, fighting both the spring and the weight of the swing leg.

## 2.7 Steering

Ranger can steer around gradual curves via a twisting degree of freedom in the inner legs (see figure 7). A small 1-watt-nominal Maxon gearmotor is connected to the inner legs via a pair of linkages, and controlled by one of the ARM7 satellite boards. For steering, the two inner leg tubes are rotated along their axis inside bearings in the inner leg box. The two inner ankles are rigidly attached to each other at the bottom, and the leg tubes themselves are quite thin and relatively flexible in bending. Thus, rotation of the two leg tubes around their own axes at the top, via the motor-driven linkage, results in rotation of both inner feet around a common central vertical axis at the bottom. By thus swiveling the feet, the robot can turn by up to about 5 degrees in one step of the inner legs. Coordination of inner leg twist with leg swing happens in the top-level processor; however, the direction and amount of this steering has thus far been controlled by a human-operated remote control.



a) neutral steering

b) steered

Figure 7: **Steering mechanism.** Steering is by twist of the inner legs. To turn the robot to the right the inner legs are turned left during the inner-leg stance phase; the feet do not slide on the ground so the robot turns. Neutral steering is shown on the left, twisted and bent legs are shown on the right. The steering motor turns the parallel linkage which twists the legs (the legs are mounted at the hips in bearings that allow axial turning). Because of the rigid link between the ankles this twisting bends the legs resulting in a turning of the pair of feet.

**Previous steering mechanism.** The mechanism described above replaces a previous ill-conceived mechanism. We had reasoned (and observed in an earlier robot) that any break in the robot's lateral symmetry would result in a turn one way or the other. So we had a motorized lever with a pulley that adjusted the length of the left outer ankle drive cable, thus changing the angle of the left foot relative to the right. This mechanism did result in some steering, but the result was highly sensitive in amount and even direction to gait, slope, and flooring material. In retrospect, the design was something like trying to steer a car by deflating the tires on one side; here we were trying to steer a robot by making it stub one or the other of its toes on the floor with each step. This steering was used for the 1 km and 9 km walks, but the 1 km walk ended abruptly when the robot was unintentionally steered over a sand pit, and the 9 km walk nearly ended with the robot heading out the back door of the gym and into the street.

### 2.8 Fall protection

The fore-aft cylinders with painted-on eyes and the foam ears (see figure 2) are for shock absorption in case of falls. The eye stalks protect against falls directly forward or backwards. Polyethylene foam pipe insulation/pool floats are guided by nylon tubes passing through holes in the top truss of the robot, and held in place by plastic caps on the ends. During a fall, the nylon tubes slide through and allow the foam to compress, absorbing the energy of the fall, but keep the foam from buckling or bending. The design was tested with a virtual robot made of wood. The final design cushioned forwards or backwards falls over a stop distance of many centimeters.

The "ears" are high-density foam rubber, intended to protect the robot in the unlikely event of a sideways or twisting fall. (As a 2D biped, Ranger has so far has always fallen directly forward or backward).

## 3 Electronics hardware

### 3.1 Evolution

Ranger's electronic hardware, its so-called nervous system, went through several iterations. The first electronics version (named the KoBrain, after its student designer) used a custom board with a Freescale MC56F8347 hybrid DSP microcontroller, external A/D converter, and three-axis MEMS accelerometers and rate gyros. A plug-in daughter board held three motor driver ICs, a wireless data communications module, and various sensor interface circuitry. The KoBrain was used for the first long distance walk attempt (1 km). At that point we realized the value of having angle sensors at all the joints, in addition to the motor shaft angle, and also wanted to install a commercial IMU (Inertial Measurement Unit). The Freescale microcontroller, though selected for its large input-output capability, was nevertheless running low on ports and processor time by this point. So three additional MC56F805 microcontroller boards were added, one per joint, communicating with the main one via CAN bus. These were bulky and not particularly energy-efficient. The third robot nervous system, described in detail below, was a network of ARM-based processor boards designed for a future biped robot with ten or more degrees of freedom. But this system was installed and debugged on Ranger.

### 3.2 Robot nervous system overview

The CAN (controller area network) bus was introduced in the automotive industry in the late 1980s to accommodate the increasing numbers of sensors and actuators in each vehicle. It allowed manufacturers to install multiple electronic control units (ECU) throughout vehicles, each handling a local subset of the sensor and control load, and communicating with each other over the high-reliability CAN network. This greatly reduced the size and cost of the wiring harnesses required; for example, masses of costly shielded low-level sensor cables could be replaced with a single twisted-pair CAN cable and a power cable. CAN also increased the modularity of the automotive electronics: ECUs from different manufacturers could be added or removed from the system with minimal impact on the rest of the electronics and support software. The needs for a robot are

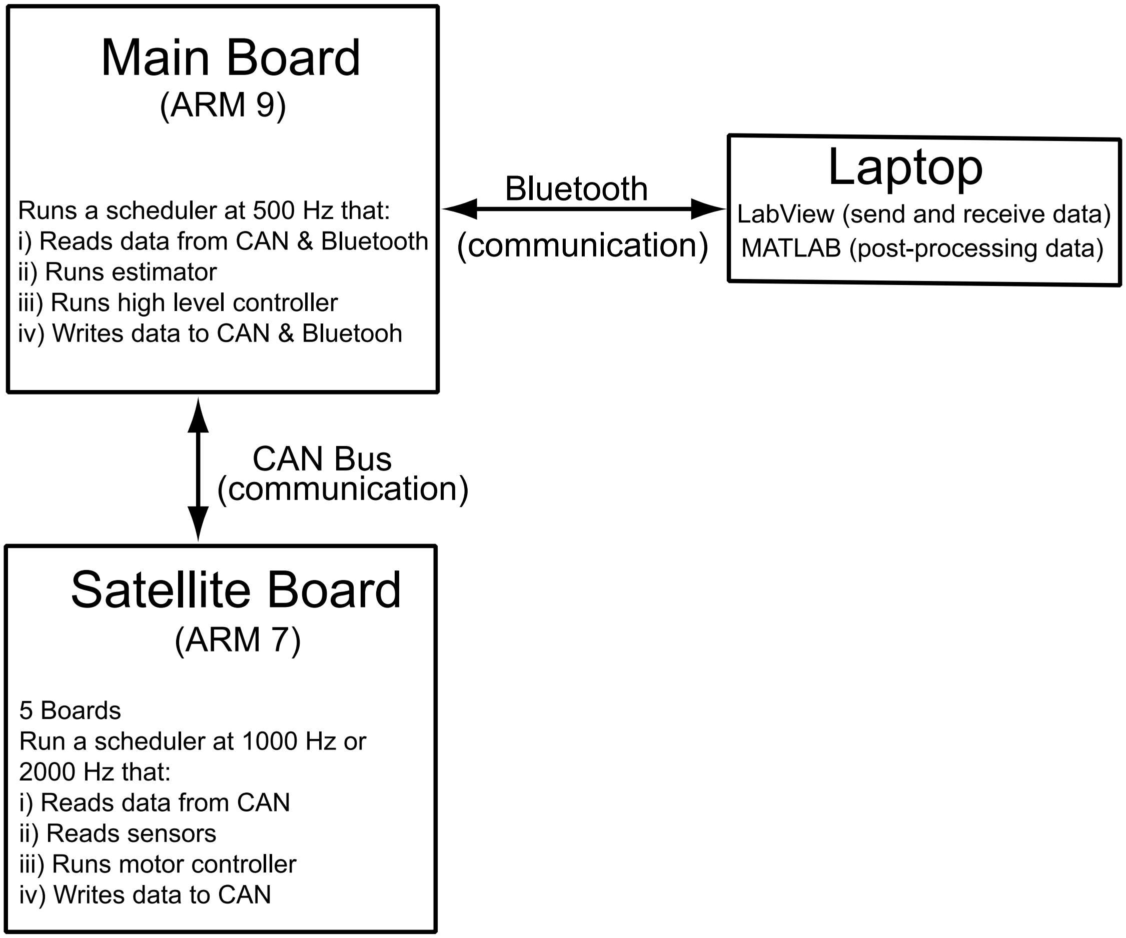

Main Board
(ARM 9)
Runs a scheduler at 500 Hz that:
i) Reads data from CAN & Bluetooth
ii) Runs estimator
iii) Runs high level controller
iv) Writes data to CAN & Bluetooh
Bluetooth
(communication)
Laptop
LabView (send and receive data)
MATLAB (post-processing data)
CAN Bus
(communication)
Satellite Board
(ARM 7)
5 Boards
Run a scheduler at 1000 Hz or
2000 Hz that:
i) Reads data from CAN
ii) Reads sensors
iii) Runs motor controller
iv) Writes data to CAN

Figure 8: **Overall electronics architecture.** An ARM 9 'main brain' communicates with 6 ARM 7 satellite processors via a CAN network. Data collection and tuning use a Bluetooth wireless data link. Not shown is the hobby radio control used for steering.

similar. Ranger uses a version of the automotive CAN bus, but with a cable and network protocol modified to work better with our small, low-power robot.

Ranger's ECUs are divided into two classes: top-level "main brain" processors based on the ARM9, and sensor-and-actuator satellite ("spinal") processors based on the ARM7. The satellite boards are connected to four high-speed CAN buses; connections between buses and to one or more top-level processors are made via a satellite board set up as a CAN router (see figure 8). The boards were designed for low energy use, and the modular design allows boards, sensors, and motors to be added or removed quite easily. Wiring complexity is reduced, since the boards are connected to each other only by power supply and CAN bus cables.

### 3.3 Top-level (main brain) processor

Ranger uses a Phytec phyCORE LPC3250 system-on-module for running high-level control and estimation. With an NXP 208 MHz ARM 9 processor and 32 MB of memory on the module, it is

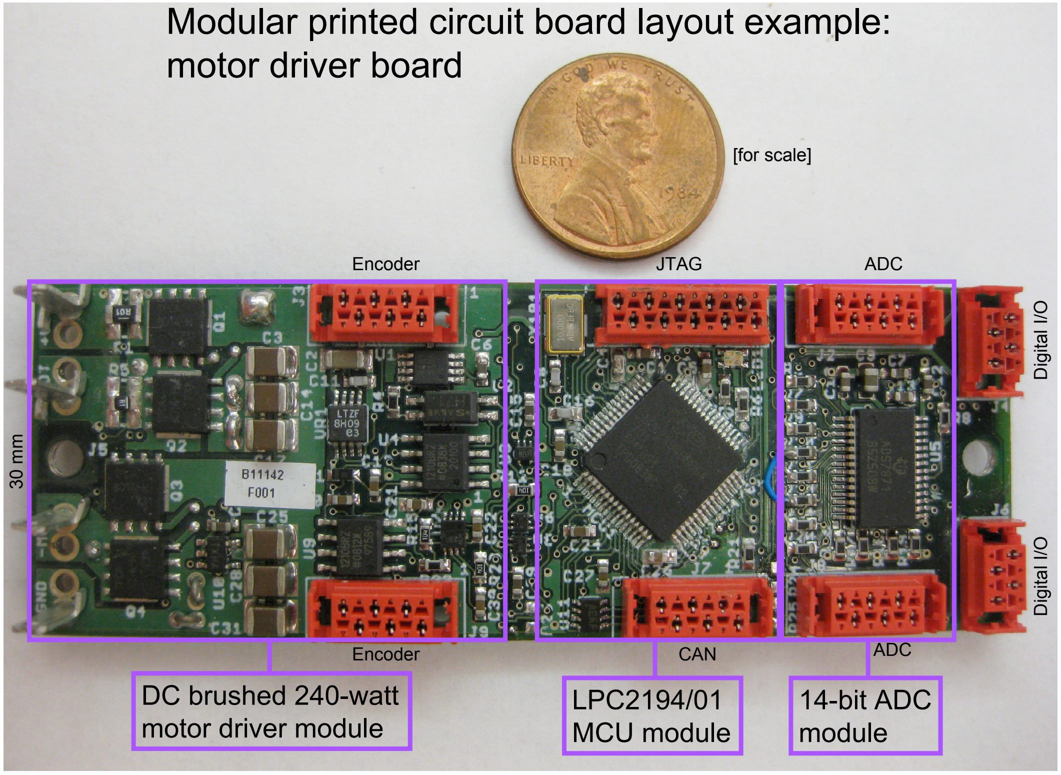

Modular printed circuit board layout example: motor driver board
[for scale]
Encoder
JTAG
ADC
Digital I/O
30 mm
Digital I/O
Encoder
CAN
ADC
DC brushed 240-watt motor driver module
LPC2194/01 MCU module
14-bit ADC module

Figure 9: **Example satellite processor board.**

clearly less computationally capable than a modern PC, but also uses only 370 mW. The LPC3250 was selected because at the time it was one of the only low-power processors available with floating point capability, which we have deemed essential for our top-level controller. At the time, available PowerPC modules used more than three watts for comparable performance, and PC-104 boards with floating point used over ten watts. The LPC3250 performance has been adequate, although has demanded some extra attention to computational efficiency.

The network architecture allows additional top-level processors to be added as needed; for example, a second processor could be added to handle sensor fusion and state estimation. In addition, other low-power boards with floating point have become available now and could be used to upgrade the top-level processor, such as the tiny Overo modules from Gumstix Computing (based on TI OMAP processors), and a variety of Intel Atom boards consuming about 5 W but giving netbook performance levels.

### 3.4 Satellite board processor

For the ARM 7 satellite boards we selected the NXP LPC2194/01 microcontroller (MCU), mostly on the basis of its low power consumption, about 80 mW at its maximum 60 MHz clock speed. Keeping the power per processor low was critical if we were to keep the power requirements of a whole network of them within reasonable bounds (Note that in the end half of Ranger's energy budget was used for processing and sensing). A second requirement was support for multiple

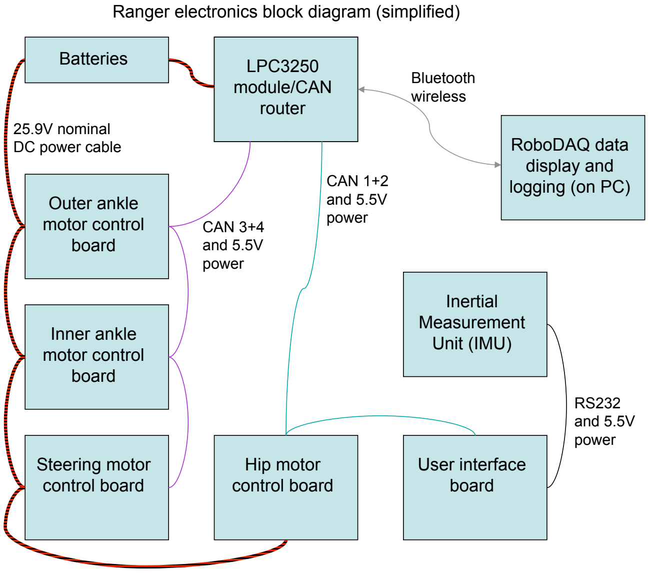

Ranger electronics block diagram (simplified)
Batteries
LPC3250 module/CAN router
Bluetooth wireless
RoboDAQ data display and logging (on PC)
25.9V nominal DC power cable
Outer ankle motor control board
CAN 3+4 and 5.5V power
CAN 1+2 and 5.5V power
Inner ankle motor control board
Inertial Measurement Unit (IMU)
RS232 and 5.5V power
Steering motor control board
Hip motor control board
User interface board

Figure 10: **Wiring between microprocessor boards.** Dark red is main power. Violet and light blue are CAN buses. The CAN router board is the port for information to and from the main brain (which is not shown).

CAN buses. The LPC2194 is identical to the better-known LPC2129, but with an additional two CAN controllers. Other peripheral support was adequate, though certainly not as extensive as the previous Freescale processors. As with the top-level processor, new higher-performance MCUs are now available that would meet the satellite board requirements, including the ARM Cortex M3 LPC17xx series from NXP and the Cortex M4-based Freescale Kinetis series.

## 3.5 Modular satellite board design

As per the network design philosophy, the satellite boards needed to be customized where necessary to the particular sensors and actuators they were expected to handle. We wanted a modular satellite board design, to allow reuse of electronics and code between the satellites whenever possible. After some initial work on a stacking board system, as used earlier on Ranger, we realized that we used far too much board area and design effort in implementing the stacking connectors. It appeared instead that we would do much better to design modules at the printed circuit board (PCB) layout stage of board development. Functional modules, for example the circuitry for the MCU and an external analog-to-digital converter (ADC), could be designed and optimized into small board layout blocks, then combined and configured to give the desired satellite board capabilities. No space is thus wasted on interboard connectors, and the flexibility of the process allows for far more diversity in satellite board design. Our PCB layout CAD program, EAGLE, does allow layout art and its associated schematics to be copied and pasted between boards, albeit awkwardly, so we used it to develop a set of layout modules. These are standardized at 30 mm height, and are intended to be placed side-by-side to form a satellite board (although they can also be dropped as needed onto larger boards).

## 3.6 PCB layout modules used in Ranger

An example module is shown in figure 9. The various modules used are listed here.

**LPC2194/01 MCU module:** Includes the MCU and other circuitry common to all the satellite boards: a pair of CAN transceivers, three voltage regulators (5 V, 3.3 V, and 1.8 V, with the last two switching for higher efficiency), JTAG programming connector, precision 10 MHz crystal, power-on reset circuit, tri-color LED, and CAN network/power connector, all on a 20 x 30 mm 4-layer board. In addition, one board layer is largely devoted to a set of parallel traces that make available all the peripheral pins of the MCU at both sides of the board, allowing connections as needed to other modules on either side.

**ADC module:** Based on the TI ADS7871 8-channel 14-bit analog-to-digital converter (ADC), the ADC module has a voltage reference and RC input filters, and is 15 mm long.

**DC motor controller module:** The brushed DC motor driver has a MOSFET H-bridge with optical isolation; sensors for battery voltage, battery current, motor current, and board temperature; two encoder inputs; a watchdog timer (to shut off the motors if the motor control software crashes); and can drive a motor continuously at 30 V, 8 A, and 100 kHz. (Maximum pulse-width-modulation (PWM) frequency is 200 kHz.) The motor control module is 40 mm long.

**System power module:** Power to each board is delivered via 5.5 V on an extra pair of wires in the CAN network cable, and stepped down from the battery voltage by a switching regulator. The system power module also has sensors for measuring current and voltage input from the battery. Only the CAN router board uses the system power module and this powers the sensors and processors on all the other boards. Motor power comes directly from the batteries, using separate, larger-gauge wires.

**Optional modules:** The layout module concept is flexible enough to allow other, larger board sizes and miscellaneous other circuitry to be added where needed, while still being able to reuse the core module layouts.

## 3.7 Satellite boards used in Ranger

The wiring layout for Ranger, as a whole, is shown in figure 10. Here is a list of the different boards used, based on the modules above.

**B2A Brushed DC Motor Control:** Each of Ranger's four motors is connected to one of these boards, which uses a motor controller module, MCU module, and ADC module placed in a line with a little space in between to allow for interconnection. The overall size is 85 x 30 mm, and it uses under 200 mW, not including the motor and MOSFETs; the MOSFET drive uses another 300 mW at 30 V and 100 kHz when active.

**B8A User Interface:** The User Interface (UI) board includes the MCU module with a 2-line LCD display, multicolor LCD driver, radio control inputs, pushbutton switches, an RS232 port, six color LEDs, and a piezoelectric speaker. The board size is about 100 x 100 mm.

**B10A CAN Router/Main Brain Carrier/Bluetooth:** The CAN router board includes the MCU module, the system power supply, four CAN ports, a Roving Networks RN-21 Bluetooth wireless module, a MicroSD port, a USB port, an EtherNet port, and connectors for plugging in the LPC3250 system module. It is the same size as the LPC3250 module, 58 x 70 mm.

## 3.8 Ranger sensors and interface hardware

Ranger has about 40 sensors and various other peripheral devices either on, or monitored by, the various satellite boards. Ideally each sensor should be on or very near a satellite board; the boards would then condition the potentially noise-prone raw sensor signal and convert it to floating-point readings transmitted onto the CAN bus. Keeping the boards in close proximity to their peripherals would reduce electrical noise problems and also reduce wiring. But because the modular electronic system was a retrofit for Ranger, we decided not to put boards at the feet, and instead ran the sensor cables up the legs. A couple of additional sensor cables cross over the top bar to connect to the logically closest board, rather than the physically closest. For example, the hip angle sensor is in the right outer leg box, but its cable runs to the hip motor control board in the left outer leg box, next to the hip motor. This was intended to simplify the programming. Unfortunately this separation also resulted, not surprisingly, in some problems with electrical noise.

Similarly, the motor control boards should be located as closely as possible to the motors. This reduces the electrical interference they cause, reduces resistive losses, and reduces the overall bulk and complexity of the wiring. In Ranger the motors are indeed quite close to their controllers. Ideally, however, the controller would be part of the motor assembly.

**Motor encoders:** Each of the three main motors is equipped with an incremental encoder on the motor shaft. The direct connection allows for simpler control at high gains. The hip motor uses a HEDS 5500 optical encoder, with 192 pulses per revolution and consuming about 100 mW. The two ankle motors use IE2 - 512 magnetic encoders, with 512 counts per revolution; they use about 50 mW each. The driver software uses timer capture inputs on the LPC2194/01 to detect transitions at the quadrature outputs from the encoders, time the transitions, and increment or decrement the position count as required by the motor rotational direction.

**Angle encoders:** Ranger has magnetic absolute angle sensors on the inner ankle assembly, both outer ankle joints, the hip joint, and the steering linkage. Except for the steering, these are based on the RLS AM8192B Hall array angle sensors, as used in their RMK3 evaluation modules. A small magnet is glued into a hole in the center of a non-magnetic shaft, and the RMK3 module is mounted over the magnet, with a small gap. The Hall sensor array detects rotation of the magnet's field, and these sensor readings are converted into a digital representation of the joint angle by the circuits in the AM8192B. The satellite board reads in encoder information in two forms:

1. Via the SSI (serial output) port on the AM8192B, which is read in by an SPI (serial peripheral interface) port on the LPC2194/01 as a binary number representing the absolute angle of the joint.
2. The AM8192B also has standard quadrature-format incremental encoder outputs, which are read in by the timer capture inputs as described above. The quadrature outputs allow

an estimated joint angular velocity to be found, along with a largely redundant relative angle reading.

On Ranger, the signals for both of these interfaces pass through a single unshielded 6-wire ribbon cable to the appropriate satellite board. RLS recommended using differential line drivers for the AM8192B outputs, but as a demo board the RMK3 didn't come with any and we didn't install any. It turned out that the output of the SSI interface in particular had quite high impedance and was thus noise-prone, despite being a digital signal. The AM8192B sensors worked well otherwise. But at about 200 mW each the 4 angle encoders make up a significant fraction of the power budget for the electronics (0.8 of 5 watts). The left outer ankle sensor and its associated satellite board were deemed redundant and, to save power, disabled for the 65 km walk.

The steering linkage uses an Asahi EM-3242 magnetic angle sensor, which has a 10-bit-resolution analog output connected to one of the ADC inputs on a satellite board.

**Battery current sensors:** We used the Maxim MAX4081S for its small size, low power, high gain (60 V/V), and high-voltage (76 V) capability. The MAX4081S IC allows bidirectional measurement of current by amplifying the voltage across a low-value current sense resistor, but can only be used on the high (positive) side of a power circuit. On Ranger the MAX4081S was used to measure the battery current going into or out of the motor control board, with good results. The similar unidirectional MAX4080S IC was used to measure current flow into the CAN router board, which powers all of the electronics except for the motors and motor drivers. The current sensor outputs go to a differential input pair on the ADS7871 ADC (or to one of the LPC2194/01's internal ADC inputs, in the case of the CAN router board.) We used current sense resistors with solid metal elements (Vishay/Dale WSLP series) because we found that the film on metal film resistors vaporized when subjected to capacitor inrush currents.

**Motor current sensors:** The motor current sensor attempts to find the current going to the motor leads by measuring the voltage drop across a low-value current sense resistor installed in series with the motor, amplifying this voltage, and then sending it to the ADS7871 ADC, which converts it to a digital value proportional to current. When the motor lead with the sense resistor is on the fixed-voltage (non-PWM) side of the motor driver H-bridge, measuring the sense voltage is not an unusually challenging task. However, when the other half of the H-bridge is driven by the PWM signal, the current sense resistor is subject to a 100 kHz AC signal at the full battery voltage. The part selected for this job, the Analog Devices AD8210, does derive a fairly clean current signal from such a noisy environment. Unfortunately, the AD8210's zero-current reading is not the same for the PWM signal as it is for DC, giving a discontinuity in the motor current readings as the PWM values pass through zero. We tried an Allegro ACS712 Hall effect sensor instead, and it worked great on the bench, but when we installed it in Ranger we saw that it wouldn't work properly when sitting next to the powerful permanent magnets of a DC motor. So instead we wrote an odd bit of code which switched the pulse width modulation signal back and forth between the two motor leads at 2 kHz, so that half the time the AD8210 was seeing a clean current signal and half the time a 30 V 100 kHz common mode signal sitting on top of it. Alternating the PWM signal between motor leads was successful in averaging out the discontinuity at zero when the motor was not spinning, but still left odd variations in current readings with zero current and a spinning motor. The AD8210 is rated to survive positive voltages of up to 68 V, but negative voltages of only -5 V. The small size of the allowable negative common-mode voltage appeared to leave the AD8210 vulnerable to certain types of voltage surges, and we had to replace two of them after using an unusual battery combination.

**Battery voltage sensor:** A resistor network that drops the battery voltage to a reasonable value, which is then buffered by an instrumentation amplifier (AD623) and sent to the ADC. The input voltage range for the AD623 was not really sufficient (the data sheet mentions this limitation in a footnote).

**Inertial Measurement Unit (IMU):** In the first version of Ranger's electronics, the custom microcontroller board included 3 axes each of MEMS accelerometers and rate gyros. However, the rate gyro output seemed quite noisy, so we added a MicroStrain Inertia-Link IMU. The angular position readings were close to what we expected from the data sheet, at 2 degrees accuracy for dynamic measurements and 0.5 degrees for static measurements, at about a 200 Hz data rate. Interestingly, the angular rate data from the Inertia-Link also looked noisy, suggesting that the original sensors were reading correctly after all and that the noise was mechanical. Later we replaced the IMU with the newer MicroStrain 3DM-GX3-25, to reduce power consumption and weight, add magnetic compass capability and achieve a higher data rate. In both cases there were some discrepancies when comparing performance with the data sheets. For the Inertia-Link, we planned for a power consumption of 450 mW because the data sheet said 90 mA and 5 to 9 V. The actual power was over 800 mW; 90 mA apparently applied to the highest possible input voltage, and lower input voltages resulted in higher currents. Similarly, the 3DM-GX3-25 advertises a "1 to 1000 Hz" data rate, without specifying that the top rate applies only to the data coming directly from the accelerometers and rate gyros; using the sensor data to estimate the angular position of the device happens much more slowly, at about the same 200 - 250 Hz of the original Inertia-Link. For its 65 km record walk, Ranger used only the rate gyro information from the IMU, and calculated its own estimates of the leg angles. By using only one rate sensor we did get a 1000 Hz data rate, but without the angular position or magnetic heading data.

IMU data was input to the satellite board via an RS-232 connection to the User Interface (UI) board. The Inertia-Link baud rate was 115 kbaud; for the 3DM-GX3-25 it was increased to 460 kbaud.

**Radio control:** For steering, an older (on hand) Airtronics AM controller was used, with two receiver pulse width outputs measured by MCU timer capture pins on the UI satellite board. Pulse width data can then be used for steering or other remote control functions. There are four timer capture pins available on the board, so a four-channel controller could be used instead.

**Foot contact sensors:** The bottom portion of Ranger's feet are designed to flex up slightly on contact with the ground, relative to the top portion attached to the ankle joint. Foot flex is measured by a Fairchild H21A1 phototransistor optical interrupter (a $2 device). Although often used as a simple on-off switch for, say, paper handling, the phototransistor versions of these devices make sensitive displacement sensors, with full scale output in fractions of a millimeter. The foot design doesn't give numerical force measurements, since its sensitivity varies with the position of the ground contact along the length of the foot, but as a ground contact sensor it has proven to be highly reliable. The output from the optical sensor goes through the axis of the ankle joint shaft and then out through a hole and up the carbon fiber leg tube. To reduce the risk of wire breakage, we used a highly flex-resistant cable, and spiraled it outward around the ankle shaft to minimize the amount of bending at any point. So far none of these cables have suffered a failure. At the satellite board, the optical sensor is powered and biased as needed, and the output is read by one of the ADC channels.

**Board temperature sensor:** A TI LM26CIM5-XHA temperature IC is mounted close to the MOSFETs on the motor controller, and set to shut down the motor driver if the board temperature exceeds 100 C. It also provides an analog temperature output voltage which goes to one of the internal ADC inputs on the LPC2194/01. We have not seen any board shutdowns due to over-temperature, and did not monitor the temperature readings in software.

**Bluetooth (and direct wire) data logging interface:** A Roving Networks RN21 Bluetooth module is soldered to the CAN router board, and connected by a high-speed serial port to the LPC3250 main processor board. The Bluetooth link (up to 230 kbaud) allows data logging and adjustment of parameters from a remote computer, generally a laptop running Windows. A hardwired connector to a second high-speed serial port (up to 921 kbaud) allows a higher data rate for tethered and bench testing. On the laptop end, a second RN21 module was installed in a custom box along with an FTDI EVAL232R (FT232RL evaluation board) RS232 to USB converter. (The external USB to RS-232 converter is required because Windows internal serial ports, when available on a laptop, do not seem to support baud rates above 115 kbaud.)

**USB and Ethernet ports:** The LPC3250 “main brain” module supports 12 Mbps USB and 10 Mbps Ethernet, so the necessary connectors and support circuitry were added to the CAN router board for possible future use (but have not been used on Ranger yet).

**Two-line LCD display:** A CFAH0802ZYYHJP display (2 lines of 8 characters each) from CrystalFontz is mounted on the UI satellite board and driven through a parallel port connected to MCU digital input-output lines.

**Piezo buzzer:** The buzzer/speaker was driven by a PWM output from the MCU on the UI board, and programmed to beep when errors occurred. Some common errors received their own distinctive alert sounds, and one enterprising student also programmed it to play songs on command, including Cornell’s theme “High above Cayuga’s Waters” which played once per kilometer during the 65 kilometer walk.

**Keypad:** A six-button keypad, read by the MCU on the user interface board, allows Ranger to be set into various states, including for example “walk,” “calibrate,” and “standby.”

**Color LEDs:** The UI board also has six tri-color LEDs powered by a Maxim MAX6966 LED driver IC, and controlled via the serial peripheral interface (SPI) on the user interface board MCU. They can be lit up in all different colors, and were used as an easy way to identify Ranger walk controller states, to detect errors from a distance, and to monitor the steering control actions.

## 3.9 Motor controller design

Ranger's motor controllers went through three major changes, roughly corresponding to its 1 km, 9 km, and 65 km walks. These changes led to substantial energy savings. Initially we used the ST VNH2SP30-E motor driver IC, which was intended for things like power windows in cars. It was small, easy to use, and had a high current capability (30 A), but was limited to 20 kHz switching frequency, 16 V maximum, and only switched one end of each half-bridge.

After the 1 km walk we noted that with a higher battery voltage we could get better performance from the nominally 12 V motors. We also noticed that our power losses in the motor windings, due to the ripple current at 20 kHz, were very high. Several motors burned out at average currents that were within the rated maximum, but which created excessive heat buildup due to the high RMS current values resulting from ripple current components. High ripple currents indicate that the pulse-width modulation (PWM) frequency is not high enough for the motor. The motor-controller system has a characteristic inductance and resistance, mostly from the rotor windings, but with small contributions from the controller switch transistors and connecting wires. The motor and controller can be modeled as a series R-L circuit, with a time constant of $L/R$; given some initial value and no external applied voltages, the motor current will decay to zero at time constant $L/R$. One simplistic measure of whether the PWM frequency is high enough is by comparing the time constant $L/R$ to the period of the PWM signal. If the PWM is fast enough, the motor current will not have time to decay appreciably before the next PWM pulse comes along, and the ripple current will be low.

The Faulhaber 2657W012CR is rated for 0.7 Ω terminal resistance ($R$) and 95 $\mu$H winding inductance ($L$). The motor driver "on" resistance is very low, as is the wire resistance. If the loop resistance is assumed to be 0.8 Ω in total (allowing 0.1 Ω for the wires and controller switches), the resulting time constant is $L/R = 120$ $\mu$s, a bit more than twice the 50 $\mu$s PWM period. The ratio of $L/R$ to PWM period should be much higher than this anyway ($L/R$ greater than five times the PWM period would be a good goal), but two additional factors made it much worse. First, the Faulhaber data sheet neglected to add in the effect of the carbon brushes, which according to our later measurements increase the resistance of the total loop to about 1.3 Ω, and introduce a voltage drop of about 0.7 V. Secondly, the VNH2SP30-E only switches the low-side switches at the PWM frequency, leading to a diode voltage drop (about 0.7 V) at the high-side switches. These two voltage drops have the effect of a nonlinear resistor in series with the motor winding, and reduce the $L/R$ time constant inversely with current. For example, at the 3.1 A maximum sustained current rating for the motor, the equivalent added resistance is about $V/I = 1.4/3.1\ \Omega = 0.45\ \Omega$. As a result, the added voltage drop cuts $L/R$ to only $95/1.75 = 54$ $\mu$s, about equal to the PWM period. For lower currents the effective resistance becomes much higher, and the $L/R$ time even smaller compared to the PWM period, thus guaranteeing high ripple currents with their associated excess heat generation and wasted power.

For the second version of the motor controller electronics, high voltage high-speed drivers were used (Micrel MIC4102) with discrete fast-switching MOSFETs (Toshiba TPCA8014-H). These could use PWM frequencies up to 200 kHz. We used 100 kHz to give a ratio of $L/R$ to PWM period of about five to one (so we thought, see below) at higher winding currents. Higher PWM frequencies consume more power in driving the MOSFET gate and output capacitances, and reduce the resolution of the PWM duty cycle. (Because of the way the PWM timers work, the duty cycle resolution is at best the PWM frequency divided by the timer clock frequency, here equal to 60 MHz. The resulting resolution equals PWM frequency/60 MHz = 0.17

At a 100 kHz PWM frequency the motors did work more efficiently, and this frequency was used for the 9 km record walk. However, upon closer investigation and modeling, the real motor power consumption still did not match what we expected from the simulation. Finally we found that the winding inductance was simply not as high as expected from the data sheet; the motor current waveforms could only be explained by much lower inductance values. Measured on an Agilent 4284A LCR meter, the measured motor inductance dropped from 94 $\mu$H at 1 kHz to 35 $\mu$H at 20 kHz, 20 $\mu$H at 100 kHz, and 16 $\mu$H at 200 kHz. Thus, the loss in inductance largely cancelled out much of the efficiency gains we had hoped for by increasing the PWM frequency. As a check on our equipment and for comparison, we also measured several brands of commercial inductors, which did not show such an inductance drop over our 1 to 1000 kHz test frequency range. We also tested the windings of a Maxon 311536 brushless motor; the Maxon motor inductance was rated at 31 $\mu$H, but in our measurements it dropped to 25 $\mu$H at 100 kHz and 21 $\mu$H at 200 kHz. So inductance drops at higher frequencies are not limited to the Faulhaber motor. We are not clear on the mechanism causing the drop, but suspect parasitic capacitances in the motor winding structure. A consequence of the declining inductance is that increasing the PWM frequency is not a productive way to improve motor efficiency even if the controller can operate efficiently at high PWM frequencies. One common approach to solve this problem and improve efficiency when driving low-inductance motor windings is to add external inductors. For Ranger, we added, in series with each motor, a 47 $\mu$H Vishay/Dale IHLP6767GZER470M11 miniature surface mount inductor with an 8.7-amp maximum current and 0.04 $\Omega$ series resistance. The added inductance brought the excess winding power losses down dramatically, brought the measured power consumption much closer in line with the simulated values, and significantly improved the energy efficiency between the 9 km and 65 km walks.

The motor driver ICs receive a pair of electrically isolated (Analog Devices ADUM1210) PWM signals generated by the PWM outputs of the LPC2194/01 MCU. They also receive a pair of isolated low-side-switch enable signals from MCU digital output lines. These allow the low-side MOSFETs to be shut off at low output currents to save power, if desired (To keep the controller simple, the low-side MOSFETs were always enabled on Ranger). For safety, a watchdog timer monitors another of the MCU digital outputs, and shuts off the power to the motor driver circuitry within about 10 ms if it doesn't see a state transition on the line in that period of time. The digital output line for the watchdog timer, for correct operation, must be toggled by a software function called by the motor control software. Thus, if the motor control software crashes or otherwise stops running, the output line becomes stuck in a fixed state, the timer runs out, and the motors are shut off. Without a watchdog feature, the motors can turn on unpredictably during programming and debugging, resulting in damage to the hardware and possibly to nearby humans.

### 3.10 Batteries

Ranger uses lithium-ion batteries based on the 18650 cell used in many laptop batteries. The initial version of Ranger used two 11.1 V, 53-watt-hour-nominal batteries in parallel, for a total of about 106 watt hours. However, the voltage of a lithium-ion battery drops substantially as it discharges, from (for an 11.1-volt-nominal battery) a maximum of 12.6 V down to a minimum cut-off voltage of 9 V. To maintain consistency in the walk controller actions regardless of battery voltage, the motor controllers were compensated to adjust for the drop. The result of the compensation was that the motors always performed as if they were operating at the 9 V minimum voltage, which was not sufficient to give full performance from the 12 V rated motors.

Therefore, with its new higher-voltage motor control boards, Ranger also was fitted with higher-voltage batteries. Initially there were four of these 25.9 V, 67-watt-hour-nominal batteries, totaling about 270 watt hours. Three were strapped to the front of each of the three leg boxes and the fourth to the back of the left leg box; all were positioned to balance the dynamics of the inner vs. outer leg pairs. This battery configuration was used for the 9 km record walk. For the 65 km distance walk, seven of the 25.9 V batteries were used, for a nominal 470 watt-hours of total charge. The battery on the back was removed, and a pair of batteries was added to the right and left outer hips, once again adjusting the positions carefully to match the dynamics of the inner and outer leg pairs.

All of the batteries were from Powerizer/Batteryspace.com, and came installed with overcharge, undercharge, overcurrent, and (for the 25.9 V batteries) cell charge balance circuitry. The 11.1 V batteries could be charged in parallel, but Powerizer technical support recommended against parallel charging for the 25.9 V batteries, apparently because it could disrupt the operation of the charge balance circuits. Thus the 25.9 V batteries had to be charged individually, checked by voltmeter for full charge voltage, and only then attached to the robot and connected in parallel for operation, a procedure that seems risky and was certainly cumbersome.

## 3.11 Network hardware

**Selection of a network (CAN and other alternative networks).** Ranger uses CAN bus for communications between the various satellite boards. CAN was developed for automobile network applications, and is also used in various industrial applications. It is designed for reliable, real-time sensor and control applications in an electrically noisy environment, which also makes it a good choice for a robot. CAN is a peer-to-peer network, so satellite boards can either communicate directly with each other, or transmit all data up to the main processor and receive all commands from it. Compared to Ethernet (and its real-time relative, EtherCAT), CAN is slower, but uses less power, less board space, and is easier to implement, making CAN more suitable for very small, low-power satellite boards. USB is not difficult to implement in hardware, but the network protocol is much more complex and less flexible, though also faster than CAN. CAN has the addtional advantage of allowing many nodes on a single cable, thus further reducing the hardware requirements (USB and most Ethernet uses point-to-point wiring; active hubs are needed to connect more than two devices.) Robots in other labs have used networks based on I2C, SPI, RS-485, etc., but these suffer from a relative lack of integration of their network protocol into the controller peripheral on the microcontroller IC. CAN controllers are highly integrated, and include such useful features as CRC error detection, bus arbitration, frame buffers, etc., making transmission of a data frame as easy for the programmer as putting it in the controller transmit buffer; similarly, a received frame can simply be read from the receive buffer when it arrives.

**CAN bus overview.** At the hardware level, a CAN bus is made up of a single length of cable, with one pair of conductors used as a differential pair. The differential nature of the signal helps greatly with noise rejection, because noise signals common to both conductors are subtracted out at the receivers. On Ranger a six-conductor ribbon cable is used, allowing enough space for two CAN buses and one pair of 5.5 V power supply wires. On Ranger these are unshielded and not twisted, but shielded and twisted-pair versions are also available if more noise rejection is needed. So far, errors in CAN packets have not been a problem, though there was some evidence that the CAN bus was causing interference in an encoder cable. CAN transceivers (nodes) can be attached anywhere along the length of the CAN bus, but it is important that the length of the drop wire between the transceiver and the bus be short. Otherwise, the variation in cable impedance at the tap points can cause the signal to reflect back and possibly interfere with the main signal. On the Ranger satellite boards, the drop wire length is kept to a minimum by having the CAN ribbon cable visit each satellite board in turn. The AMP Micro-MaTch connectors we used can be crimped onto the CAN bus cable at any point, and then plugged into the board. Thus the drop length includes only the connector pins and the short board traces to the CAN transceiver IC. Adding additional boards to the bus is easy - just crimp on another connector somewhere along the cable. These new connections can't really be removed without damage to the cable, but unused bus taps can be plugged up to prevent shorts. Another benefit of using standard-size ribbon cable is the availability of compatible 3M series 3319 high-flex ribbon cable. Used to span rotating joints in the robot, it is rated for a flex life of 100 million cycles.

**CAN bus speed.** The maximum rated speed for CAN bus is 1 Mbps (bits per second). The speed limitation arises from the timing requirements of the arbitration scheme used to resolve data frame collisions, when two nodes begin transmitting at the same time. In this case both transmit together until, eventually, one node transmits a zero and the other a one. Then the zero is dominant and appears on the bus. A station transmitting a one but seeing a zero on the bus recognizes that it has lost the arbitration and stops transmitting. This arbitration scheme allows efficient and predictable use of the bus, because the higher-priority frame always gets through and no time is lost to arbitration. However, if the transmissions are not well synchronized, the arbitration process will fail.

**Overclocking the CAN bus.** Electric signals travel at about 150 meters in a microsecond (about half the speed of light). For 1 Mbps the maximum cable length to maintain synchronization is about 40 m, approximately the scale of a car wiring harness. However, for a small robot like Ranger the bus cables are only a meter or two long. In conjunction with a fast CAN transceiver IC (we used the Maxim MAX3051), the short bus length opens up potentially higher speeds. Many microcontrollers with CAN controllers built in can use higher clock rates to give higher CAN bus speeds, up to about 6 Mbps in the case of the LPC2194/01. In our tests at 6 Mbps the bus did not work no matter how short the cable, and at 5 Mbps it worked only with very short cables, under a meter. At 4 Mbps, however, the performance was good over a 10-meter cable. (We did discover eventually that the acceptance filter for automatically sorting incoming frames did not work for the LPC2194/01 at such high speeds, but it wasn't clear that it worked properly at the normal 1 Mbps speed either - there is mention in the LPC2194/01 errata sheet of problems related to the acceptance filter.)

**CAN bus data rate.** For Ranger we used four independent CAN buses each running at 3 Mbps, which worked well with the 60 MHz processor clock speed and did not push the timing requirements quite so hard. Note that using the higher CAN speeds requires a self-contained set of boards and transceivers selected and set to operate at the non-standard speed. Commercial CAN-compatible boards were not used in Ranger, but conceivably could have been if one of the buses had been set to operate at 1 Mbps. Each of our CAN frames is 108 bits long, using the maximum 8-byte data payload. Assuming that we allow a maximum bus loading of 75 percent, that means Ranger can handle at most 0.75 x 4 x 3Mbps / 108 = 83,000 frames per second. At first glance 83,000 sounds like a lot, but we want our walk controller to react to sensory information within a few milliseconds. Thus, many of the sensor values need to be sent at 500 Hz, using up the available bandwidth rather quickly. On Ranger we were able to send all of the data we wanted, but some less critical values were sent at a lower rate. For a more complex robot, we would need to pay more attention to data compression, use more CAN buses, or both.

**CAN bus layout.** The CAN Router/Main Brain Carrier board is connected to all four CAN buses; the other satellite boards are connected to two each, though in practice each board only uses one of them for communication. The CAN router's main job is receiving CAN packets from the four buses, checking its lookup table for the correct destination(s) for each frame (based on its 11-bit data ID), and sending them out again. It is the main timekeeper for the system, and for synchronization sends out a CAN frame with the current elapsed time every millisecond to all the other satellites. It also handles high-speed SPI (serial peripheral interface) communications to and from the top-level LPC3250 processor.

### 3.12 Loading code into the microcontrollers

We used the Keil uVision integrated development environment (IDE) for ARM processors, and programmed the flash memory on the MCUs via their JTAG ports using Keil's ULINK debug adapter box. For the satellites, the normal 20-pin 0.1-inch-pitch JTAG port was replaced with a 10-pin AMP Micro-MaTch connector to save space, and used with a small 10-pin adapter board that plugged into the end of the 20-conductor cable from the ULINK. The LPC3250 module has a 20-pin JTAG port onboard; this port was extended to a 20-pin jack on the back of Ranger for convenience. Programming the satellite boards requires removing the thigh box covers, but reprogramming is seldom needed after the code is running well. In principle the satellite code could be distributed automatically via the CAN bus, but so far we have programmed the satellites directly.

## 4 Software for the robot microcontrollers

Later sections of this appendix describe software used for simulation and controller development. Here we describe the software running on the robot itself. By far, this is dominated by code associated with sensing, communications, and low-level motor control. The logic that controls the walking is a small part of the total code.

### 4.1 Overview

**Main brain vs satellites.** Ranger's software architecture represents a balance between two goals: moving processing tasks down to the satellite boards whenever feasible, to improve response time, free up top-level processor resources, and minimize data traffic; and having a single, convenient block of code to work with while improving the performance of the walk controller (compartmentalizing the code). The resulting division of labor between the top-level processor and the various satellites is relatively conventional: the satellites handle high-speed motor control loops and sensor data inputs, while the top-level processor runs the main estimation and state machine code.

**Modular design.** Paralleling the modular satellite board electronics, the software for the satellite boards is also mostly modular. Each board includes central code modules which handle functions common to all; a subset of the available peripheral driver modules, selected to support the features needed for a particular board; and several setup files to tie all the code together and integrate it with hardware-specific board features.

**Data table.** Data integration and communication across the networked system of processors starts with a robot-wide master table of variables, each with a unique 16-bit data identifier. The table includes parameters that are fixed at the outset and ones that are measured and calculated, including: motor powers, currents, angles and angular rates; battery current and voltage; estimatied dynamic state variables like absolute let angles; number of steps; estimated distance travelled; and various finite state machine parameter (e.g., nominal control level, stiffness, dampness and gain constants). A publish-subscribe system is used; the value associated with any particular data id is generated and published by a single source somewhere in the system, but multiple data destinations (subscribers) are allowed. These data sources and destinations are included in the table, along with an initial value and network routing information. The table is generated as a text file offline, then parsed by a MATLAB script, which generates dedicated C code files for each of the boards in the robot, and also generates code for the data logging and monitoring program. Keeping data definitions consistent across the network is thus automated. In some cases, changes to the master variable table require that all of the code for the robot be recompiled and reloaded onto the boards. However, by adding spare variable slots in the table for each board, we usually only need to reload the code for one or two boards at each onboard update.

**Data sharing.** At run time, the top-level processor on Ranger maintains a structure containing all of the robot's data. The satellite boards only store locally relevant data. When a variable is updated anywhere in the system, it is added to the local schedule to be transmitted as needed to the boards and processes on the subscriber list.

## 4.2 High Level Control: Hierarchical Concurrent Augmented Finite State Machine (FSM)

Ranger's high-level walking control runs in a hierarchical concurrent finite state machine (FSM) on the top-level processor, written in C++. Each motor in the robot - the hip, two ankles, and one steering motor - is assigned its own state machine. These four state machines run concurrently and independently of each other, but are implicitly synchronized by the robot's global dynamic state (the shared data table). For example, heel-strike data can trigger transitions in several state machines at once. And within a state the shared (augmented) data is used in the control. The hierarchical aspect of the state machine was not used in the present controller design, but it is possible to nest one state machine within another, and thus create a multi-level state machine. For example, a "behavior" state could include both a "stand-still" state, a "start-walking" state and a "walk-forwards" state, and each of these could contain four state machines corresponding to the four motors.

**States and transitions.** Each state machine (one for each motor) is made up of a set of states, and a set of transition conditions between states. Each state includes three actions: an entry action function, which runs upon entry to the state; a main action, which runs a control action as long as the state is active, using the 'augmented' data from the table; and an exit action, which runs upon exit from the state. The entry actions and exit actions enable initialization and cleanup of temporary variables, respectively. The transition conditions (e.g., heel-strike), establish the rules for transition from one state to another.

**Compliant control.** Within a state the instantaneous control action typically involves sending a motor command value $I_{\mathrm{i0}}$ to the motor controller at the target joint. This value, nominally a current, is modified by two additional parameters sent previously (possibly in the state's entry action), a proportional gain $K_{\mathrm{ip}}$ and a derivative gain $K_{\mathrm{iv}}$, such that the mechanical impedance at the joint can be controlled. The desired actual motor current $I_{\mathrm{ia}}$ is calculated at the satellite board as:

$$I_{\mathrm{ia}} = I_{\mathrm{i0}} + K_{\mathrm{ip}}\theta_{\mathrm{ia}} + K_{\mathrm{iv}}\dot{\theta}_{\mathrm{ia}} \tag{3}$$

Here $\theta_{\mathrm{ia}}$ and $\dot{\theta}_{\mathrm{ia}}$ correspond to the motor position and motor velocity of the joint that is being controlled.

## 4.3 Sensor fusion and state estimation

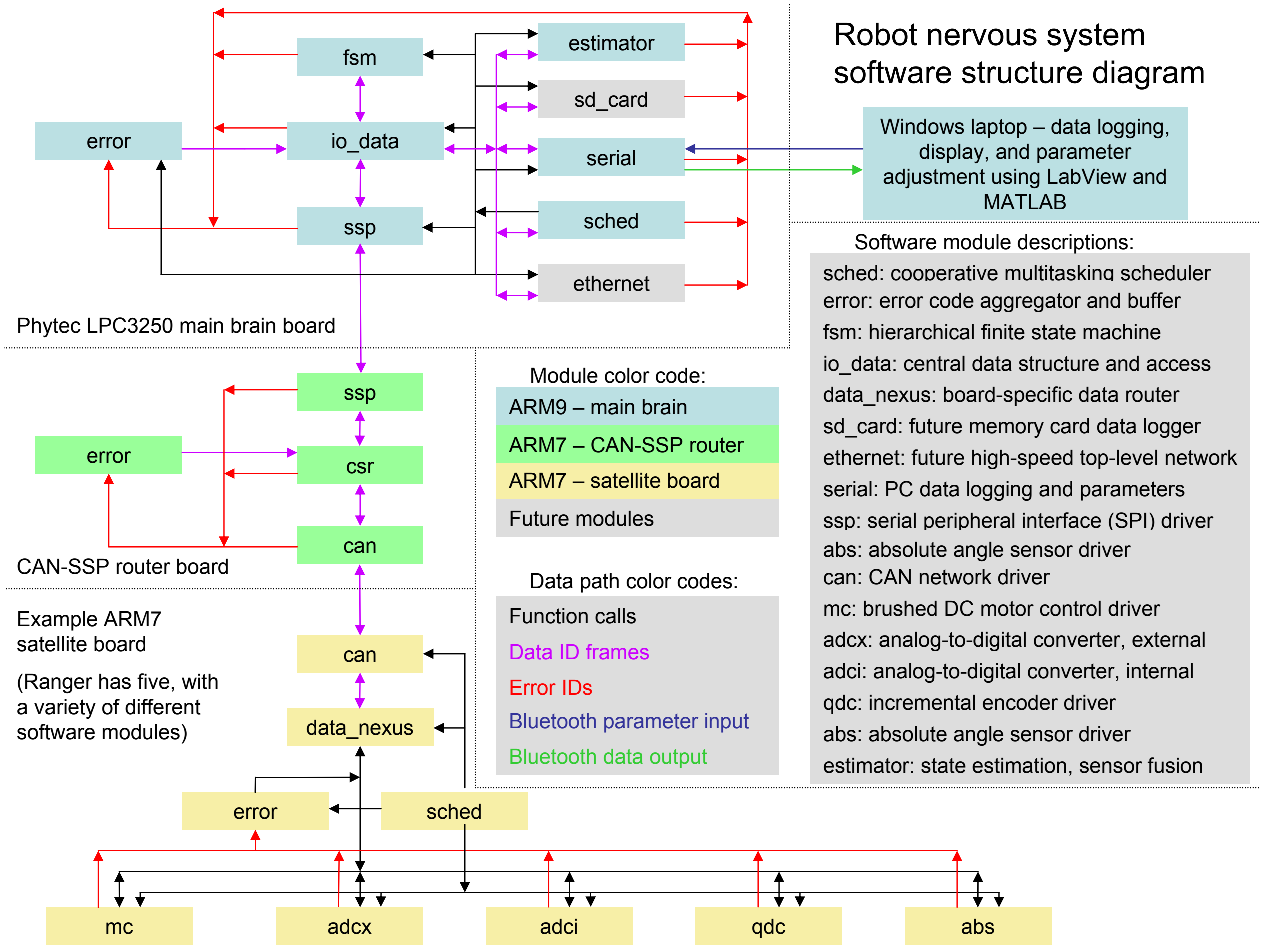
The state estimation module on the main brain tries to determine the dynamic state of the robot. It determines which foot is on the ground and the absolute angle and angular rate of the stance leg. The estimator code also keeps track of step time, step velocity, step length, number of steps, distance traveled, etc.

**Dynamic state and heel-strike timing.** Ranger's control depends on knowledge of the absolute angles of the legs relative to the direction of Earth's gravitational acceleration. Especially important is the height of the swing foot from the floor so as to prevent toe-stubbing and to control the step length. These measurements are not directly available from Ranger's onboard sensors, and must be derived via sensor fusion. The MicroStrain IMU (Inertial Measurement Unit) does have an internal general-purpose sensor fusion algorithm for determination of absolute angle, but the data rate of the rate sensor when used alone is higher, and our sensor fusion, based on ground contact and robot kinematics, is more accurate.

On Ranger we use an algorithm which determines absolute leg angles by fusing the sagittal plane angular velocity data from the IMU with other onboard angle and contact sensor data. At heel-strike the configuration of the robot relative to an assumed flat level floor is fully known from joint angle data; at that point the angular rate integrator is partially reset to reflect the newly-measured robot pose relative to the floor, and integration continues from that point forward through the step to the next heel-strike. The degree of reset per step was selected to give a reasonable balance between errors in measurement at heel-strike, roughness of the floor, and the drift rate of the angular velocity measurement. For Ranger, the integrator reset is 10% per step. That is, just after heel-strike the estimate of angle of the IMU is taken as

$$0.9 \cdot (\text{value from integration from last step}) + 0.1 \cdot (\text{value from joint angles assuming level floor}).$$

Note that the state estimator in its current form is not doing 'model-based-estimation' of the type


Robot nervous system software structure diagram
fsm
estimator
sd_card
error
io_data
serial
ssp
sched
ethernet
Windows laptop – data logging, display, and parameter adjustment using LabView and MATLAB
Phytec LPC3250 main brain board
ssp
error
csr
can
CAN-SSP router board
Module color code:
ARM9 – main brain
ARM7 – CAN-SSP router
ARM7 – satellite board
Future modules
Data path color codes:
Function calls
Data ID frames
Error IDs
Bluetooth parameter input
Bluetooth data output
Example ARM7 satellite board
(Ranger has five, with a variety of different software modules)
can
data_nexus
error
sched
mc
adcx
adci
qdc
abs

Software module descriptions:

sched: cooperative multitasking scheduler
error: error code aggregator and buffer
fsm: hierarchical finite state machine
io_data: central data structure and access
data_nexus: board-specific data router
sd_card: future memory card data logger
ethernet: future high-speed top-level network
serial: PC data logging and parameters
ssp: serial peripheral interface (SPI) driver
abs: absolute angle sensor driver
can: CAN network driver
mc: brushed DC motor control driver
adcx: analog-to-digital converter, external
adci: analog-to-digital converter, internal
qdc: incremental encoder driver
abs: absolute angle sensor driver
estimator: state estimation, sensor fusion

Figure 11: **Overall organization of onboard software.**

of common observers or Kalman filters. But the integration reset at each step is a simple kind of model-based estimation. This algorithm, using the angles determined above and the geometry of the kinematic chain knowing the stance foot was on the ground, could measure swing foot height to an accuracy of about 2mm. Our lab floor has step to step variation that is also about 2mm. Hence on the lab floor the swing foot collision could be anticipated with a typical accuracy of about 3-4 mm. As discussed later, accurate prediction of the time of heel-strike based on this dynamical state estimation is enhanced by the foot having a large vertical velocity at heel-strike.

**Finding the time of heel-strike.** The accuracy of the angle estimation (above) depends on the accuracy of the heel-strike detection algorithm; any error in the estimate of the time of heel-strike translates into errors in the estimated absolute leg angles. Heel-strike is detected by ground contact sensors on Ranger's feet. These generate an analog voltage roughly proportional to the force on the heel from the ground. This voltage can have a variety of errors, including high-frequency electrical noise and drift due to temperature, creep, etc. Identifying heel-strike using the unfiltered voltage and waiting for only a small voltage change could lead to spurious detections. On the other hand, excessive filtering, or waiting for a large change in the force, adds delays to the detection time. Although retrospective detection is fine for integration of angular rates, the controller also has a logical-state transition at heel-strike; delayed detection of heel-strike is problematic for push-off. The heel-strike algorithm on Ranger uses the following algorithm:

1. The swing feet contact sensor voltages are averaged for a period of time during each step while the feet are in the air, well before heel-strike is expected, thus setting up a baseline for comparison and removing long-time sensor drift errors. This zeroing is done while the swing leg is forward of the plane of the hip and the foot has not yet flipped down for heel-strike.
2. Starting from when the foot is flipping down into position for heel-strike, sensor voltages are measured relative to the just-determined baseline. If this baseline voltage for one of the feet is outside of the normal range, that foot's sensor is considered stuck or otherwise in error and ignored for that step.
3. When the contact sensor voltage rises above a predetermined threshold and stays there for at least 2 ms, on either foot, heel-strike is declared and the time is recorded.

The average heel-strike detection delay (time difference between an actual heel-strike and its software detection) is 14 ± 3 ms. This delay could be reduced if the threshold were decreased (or the 2 ms limit were decreased). But both of these would reduce the reliability of the detection: a spurious noise or a scuff on debris could cause a false detection of heel-strike.

**Angles and Angular rate estimates.** The angle sensor data from the joint angle sensors is fairly clean (e.g. 0.0008 rad resolution for hip angle, and 0.001 for ankle angles), but the angular rate data is quite noisy. The estimation module filters the local angular rates from the joints and the absolute angular rate from the IMU (which provides a measurement of the absolute angular velocity of the outer leg pair, to which it is attached). A second order Butterworth filter with a cut-off frequency of 10 Hz is used for all of the rate data. A Butterworth filter was chosen because of its maximally flat characteristics in the pass and stop bands. A cut-off of 10 Hz was chosen by looking at the frequencies of the prevalent noise (fluctuations) in the data. The filtering process introduces a phase lag (delay) between the raw and the filtered signal. As always for a filter of given order, the delay and 'smoothing-ability' work against each other. The delay for the current filters is about 20 ms. The filter coefficients were tuned manually by judging delay versus smoothing capability on a test sample of data. Also note that the joint angle rate data received in the main brain has already been filtered in the satellite using a first order filter and hence there is some additional delay of about 5 ms.

## 4.4 Low-level software overview

**Operating system.** The main processor and satellite boards do not run any commercial operating system. Instead they rely on a simple cooperative-multitasking scheduler function. All of the low-level code is written in C, except for a few peripheral drivers written in assembly language. The main processor scheduler runs at 2 ms intervals (500 Hz); the motor controllers run at 0.5 ms intervals (2 kHz); and the other boards run at 1 ms intervals (1 kHz).

**Scheduling.** The order and selection of functions to run in any one of these intervals (schedule slots) is determined by a table. Functions can be placed in one of three categories: 1) Run in every slot. 2) Run in one or more slots (rows) of the table, which is typically ten slots in length. 3) Run occasionally - one function from a list is selected to run during each pass-through of the whole table. These options allow the run rate of a function to be set approximately as desired, from the full control loop rate down to 10 Hz or less. One shortcoming of our system, compared to a commercial real-time operating systems (RTOS), is that all of the functions listed for a slot are required to complete before the end of the interval, putting a strict limit on the amount of processing time available. For example, even if a function on the motor control board only runs once every 100 ms, it still needs to complete in a small fraction of 0.5 ms. Guaranteeing that all the functions complete prior to the end of a slot requires that no function can contain an endless loop or wait for an external event to occur. Instead, each function checks to see if there is anything it can or needs to do, does it quickly, and exits. Execution of the functions in each slot is timed, and approaching or exceeding the allowed slot time generates an error message.

**Interrupts.** High-speed sensor and control input-output (I/O) are handled by interrupt functions at a variety of priority levels, a standard MCU feature which allows normal scheduler operation to be stopped temporarily in order to run a time-critical peripheral function instead. Normal scheduler operation resumes once the interrupt function finishes. As an example, Ranger's ankle motor shaft encoders can send out pulses at over 100 kHz; to avoid miscounting or mistiming these pulses, the appropriate interrupt function must be called immediately each time a new pulse arrives, and thus it receives the highest priority level. CAN frames can arrive at rates of up to 25 kHz per bus, also requiring a high-priority interrupt.

For the top-level processor, the general order of operations during each scheduler slot is:

1. Read in all pending data from the satellites and Bluetooth into the main data structure.
2. Run the state estimator functions.
3. Run the walk controller.
4. Write all command data to the main data structure, for transmission to the satellite boards.

For the satellites it is similar:

1. Run sensor data update functions.
2. Read incoming CAN data.
3. Run the motor controller (for example).
4. Transmit data on CAN bus (sensors, error codes, etc.)

## 4.5 Code module overview

The various boards differ in their application and sometimes their capabilities, but they share core scheduling and network communication functions, and many of the peripheral functions too. These are all encapsulated in reusable code modules, with rules governing their interfaces with other modules, the setup code, and the electronics hardware. Code module software configuration, data input, and data output are all via a set of public function calls, a fairly standard approach to modular code organization. With the interface to the electronics hardware, we make the connection between the software functions and the details of the electronics as clear and straightforward as possible. Rather than trying to abstract away the details of the hardware interface, our approach does the opposite. For example, MCU peripheral setup registers are referred to by exactly the names given to them in the manufacturer's user manual, for easy reference. Register bits set or cleared are easily identified and looked up if needed.

**Code modules.** Each code module is comprised of a set of functions, some for in-module use only (private) and some intended to be called by other modules (public). A generic set of such functions would include:

Software initialization functions, to be run once at MCU startup.

Input functions, to write data to the module

Output functions, to read data from the module

Update functions, running at predefined intervals. This is where the main processing happens, if needed.

Interrupt function, for peripheral modules with tight timing requirements.

Error logging functions.

In addition, code modules working directly with hardware need this hardware to be initialized properly. Each module has a section at the top with example code for setup of the interrupts and other hardware items it needs. Although that example code doesn't run, it can be copied and pasted as needed into board-specific hardware and interrupt setup functions. These functions take care of the setup for all of the hardware on the board, and run at startup along with the software initialization. By keeping the setup all in one place, it becomes much easier to coordinate the hardware resources between the various code modules, and to avoid conflicts.

## 4.6 Quadrature decoder(QDC) module

Incremental encoders are displacement sensors that output a sequence of electrical pulses in proportion to their relative motion. For example, the Ranger ankle motor encoders send out 512 pulses per revolution of the motor shaft, or approximately one per $360/512 = 0.7$ degrees. A single stream of pulses can tell us how far and how fast the shaft has moved relative to its starting position, but not its direction or absolute position. For incremental encoders, the direction problem is solved by adding a second channel of pulses, out of phase with the first by 90 degrees but otherwise identical (this is called quadrature encoding). When the phase of output A leads that of output B, the shaft is spinning one way, and when the phase of A lags B it is spinning the other way. The QDC module is set up to interrupt when (for example) channel A undergoes a transition in either direction. Based on the state of channel B at the time of the transition, the code can determine the direction of the shaft and thus whether to increment or decrement the position counter. In addition, it can estimate the speed of the shaft by timing the interval between the transitions. Note that the QDC code counts each cycle of channel A twice, once on the rising edge and once on the falling edge,

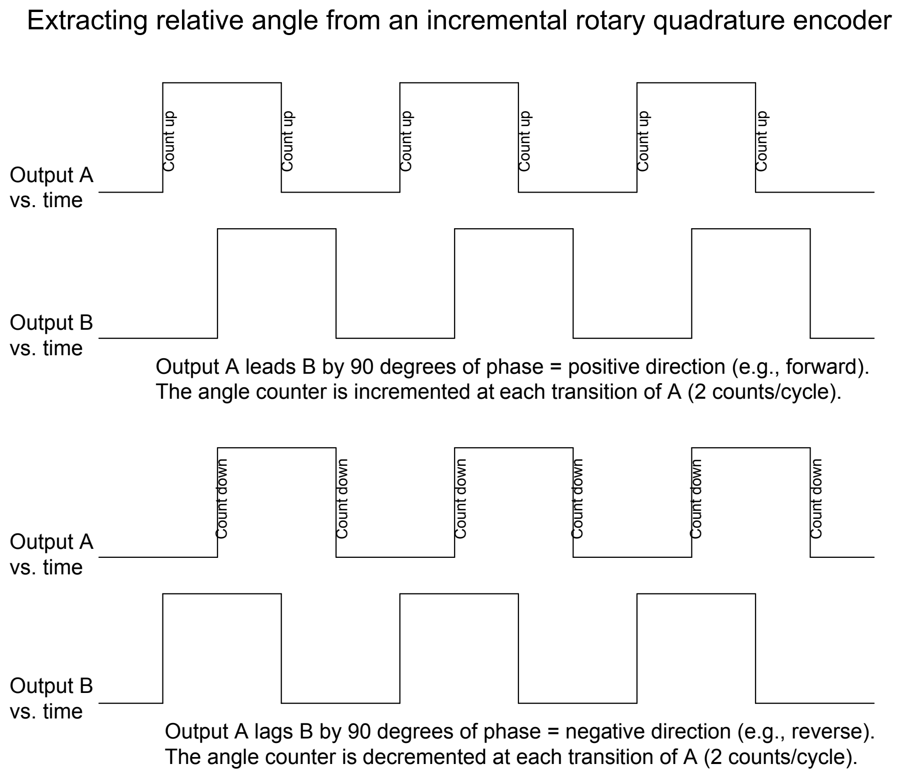

Extracting relative angle from an incremental rotary quadrature encoder
Count up
Output A vs. time
Output B vs. time
Output A leads B by 90 degrees of phase = positive direction (e.g., forward).
The angle counter is incremented at each transition of A (2 counts/cycle).
Count down
Output A vs. time
Output B vs. time
Output A lags B by 90 degrees of phase = negative direction (e.g., reverse).
The angle counter is decremented at each transition of A (2 counts/cycle).

Figure 12: **Extracting relative angle from an incremental rotary quadrature encoder.**

giving 2 x 512 or 1024 counts per revolution (2X counting). By also counting the rising and falling edges of channel B this could be doubled to 2048 counts per revolution and four times the base resolution (4X counting). However, the code to do 4X counting requires more processing per count, and has twice as many counts to process as does the 2X counting for a given motor speed; processor speed is an issue.

**Encoder decoding code.** The encoders generate pulses at a high rate when the motors are spinning fast. For example, when an ankle motor is spinning at its full speed, 6300 RPM, the driver software receives pulses at a rate of 6300 * 512 / 60 = 54 kHz, and the interrupt must fire twice per pulse, at 108 kHz. The processor is operating at a clock speed of 60 MHz, so there are (at most, if the processor did nothing else at all) 60 MHz/108 kHz = 558 computation cycles per call to the encoder interrupt function available if it is to keep up with the motor and not lose count. In fact, early versions of our driver did slowly lose count, causing Ranger to fall after a few hundred steps. As a result, code was added to reconcile the ankle motor encoder readings gradually with the respective joint angle sensor readings, preventing the error from growing. In addition, the drivers were rewritten in ARM7 assembly code, reducing the clock cycle count per interrupt function call from about 500 down to about 150. This code optimization lets the motors run at full speed without interfering as much with other running code. In fact, the encoder zeroes no longer seemed to drift with time after the hand coding. Nonetheless, the code to reconcile the incremental-based and absolute angle sensor readings was left in place, just in case.

**Measuring motor velocity.** The encoder interrupt function, besides incrementing or decrementing the position count with each pulse, also sends back the number of clock pulses since the previous encoder pulse. Knowing the pulse period gives us a way to estimate the angular velocity of the motor in addition to its position, but there are some difficulties. First, the encoder pulse widths are not uniform, but have large variations pulse-to-pulse. Second, the number of pulses per second (and thus the data rate) drops as the RPM drops, and becomes zero when the motor is stopped. We wrote a velocity-estimation function that keeps an average of the most recent pulse time values and uses these to estimate the motor velocity. If the pulses stop (very slow or stopped shaft), a second timer keeps track of the time since the most recently received pulse, allowing the function to generate an ongoing estimate of the shaft speed even as it comes to a complete stop.

## 4.7 Motor controller module

**Current control.** At the satellite level, motor control code is built as two nested control loops. The inner loop controls the motor current (a common practice in motor controllers), based on the requested current from the outer loop and feedback from the sensors. At present on Ranger the current controller uses a proportional-integral (PI) control loop, with the emphasis on the integral portion to minimize overshoot. A controller incorporating an additional open-loop (model based) term based on motor speed and battery voltage was demonstrated to give faster response times, but is not yet in use.

**Compliant control.** The outer of the two control loops is a motor compliance controller, using three different control parameters from the main processor to establish a motor current to send to the current controller. The motor current target value is generated as a linear combination of three input parameters: A current offset, a coefficient multiplied by the position, and another coefficient multiplied by the velocity (See Eqn. 3). Compliant control allows the motor controller to behave as a current/torque controller, a position controller, a velocity controller, or any desired combination of the three, through appropriate selection of the three control parameters.

**Coupled motor limits.** Ranger's motors are rated for a 6300 RPM no-load speed at 12 V; i.e., at 12 V and 6300 RPM they generate no output torque. However, with a higher battery voltage, they can generate substantial torque at this speed. A voltage above 12 applied with no load, though, will make the motors spin too fast, and when applied at low speeds may overheat the motors. Thus, separate torque and speed protection functions are required, one keeping the current within safe bounds regardless of the input voltage or speed, and another limiting the speed to a safe value. Moreover, it may be desirable to temporarily drive a motor at currents above the maximum sustained value; short bursts of high current can be used safely if the rotor temperature is not allowed to exceed safe values, and simultaneously, if the peak torque is limited to values that will not cause immediate mechanical damage to the motor or gearhead. For Ranger, therefore, the current limiter has two parts. One incorporates a simple thermal model of the motor armature, based on the rotor thermal time constant and allowed maximum sustained current as given in the data sheet. The measured motor current passes through an exponential averaging filter with a time constant set equal to that of the rotor, and the resulting average is used as the basis of comparison for the thermal limiter. A second limiter looks at the instantaneous (or very short time constant) value of the current, and limits it to give no more than the maximum intermittent torque specification listed on the gearhead data sheet.

## 4.8 Central data module (IO Data)

System-wide parameter and variable data is maintained in a data structure on the top-level ARM9 board. Each variable or parameter to be included is assigned a unique 16-bit data ID for reference. The set of possible data IDs includes but is not equal to the set of CAN network IDs, because the standard CAN ID used in Ranger is limited to 11 bits (actually only the first 2032 of the 2048 possible values in 11 bits, due to CAN implementation details). Higher-value data IDs can be used in the top-level processor, but cannot be transmitted over the CAN network. Each data point in the structure includes a 32-bit data value and a 32-bit time stamp value measured from turn-on time (limiting our walks to 50 days duration). Access to the data structure is through a variety of set- and get-value functions. While the time stamp data type is limited to 32-bit unsigned integer, a union substructure allows the data values to be accessed as any of several data types, so far including single-precision floating point and 32-bit signed and unsigned integers. An additional optional field allows the module to keep track of whether or not subscribing processes have received the latest changes to a data point. This field makes it possible, for example, to limit the data that goes over Bluetooth or CAN to just the data values that have changed, instead of wasting bandwidth by sending a steady stream of identical values. Rather like e-mail, each data value can be marked as read or unread by each subscribing (local) process. There are plenty of additional features that could be added to Ranger's data and network protocol, including the addition of a guaranteed-delivery protocol. One such was largely written, but the data loss rate on the network has been so low that setting this protocol up on Ranger was not a priority. Data types longer than 32 bits would also be good to have, but a large majority of the data values on Ranger are 32-bit single-precision floating point numbers.

## 4.9 Error handling modules

If we want to have clear, reliable data from testing a robot controller, we need to know that the robot is actually running the control code we think it is. It is useless to run tests if the robot is, unbeknownst to us, actually in an error state and thus effectively running a different controller. The error handling modules address this issue. As with the system-wide data IDs described above, the robot also has a system-wide list of 16-bit error codes; these also have a MATLAB script to generate consistent C code for all the boards in the system. When a function in one of the code modules detects an error, it calls the error-occurred function from the error module with the unique error ID number for that condition. The error module maintains a buffer of error messages that been reported so far. Upon receiving another error message, it searches the buffer to see if this is a new error, or a recurrence of an earlier error condition. If new, the error is added to the buffer; if a repeat error, the matching 10-bit error frequency field is incremented. Periodically (as per the scheduler table) an error code is popped off the buffer and transmitted over the CAN bus, receiving as it goes a 6-bit board identifier field and a 32-bit time stamp. The resulting error frame travels over the CAN network to the top-level processor, and then to our data logging and monitoring program (RoboDAQ, described below) on a laptop. Error conditions at the top-level processor are handled similarly, though of course they need not go through the CAN network. RoboDAQ maintains a list of all the detected error conditions, their descriptions, and their total number of occurrences on its error tab, sorted by most recent. It also logs them to the hard disk, as with other received data, and displays the most recent error description and a (virtual) red light on the main data display page.

### 4.10 Foot contact sensor code

This is the low level code used fed to the state estimator for foot contact. Each of Ranger's feet has an integral ground contact sensor based on optical detection of foot structure deflection (described in more detail in the hardware section). The output of the optical sensor is an analog signal which is converted to a voltage measurement in the analog-to-digital converter (ADC). Each 0.5 ms, a new measurement is compared to two threshhold values; when 10 consecutive measurements exceed the upper threshold, the ground contact signal for that foot goes high (true); when the measurements drop below the lower threshhold for ten consecutive times, the foot's ground contact signal goes low (false). This hysteresis and debouncing procedure is intended to keep sensing fast, while minimizing the chance of a false heel-strike detection due to mechanical or electrical noise. At the top-level processor walk controller, the foot-contact signals coming from each pair of feet are logically read such that foot contact on either foot will be interpreted as a heel-strike for the purposes of the walk-control state machines.

### 4.11 RoboDAQ data acquisition program and Bluetooth communications module

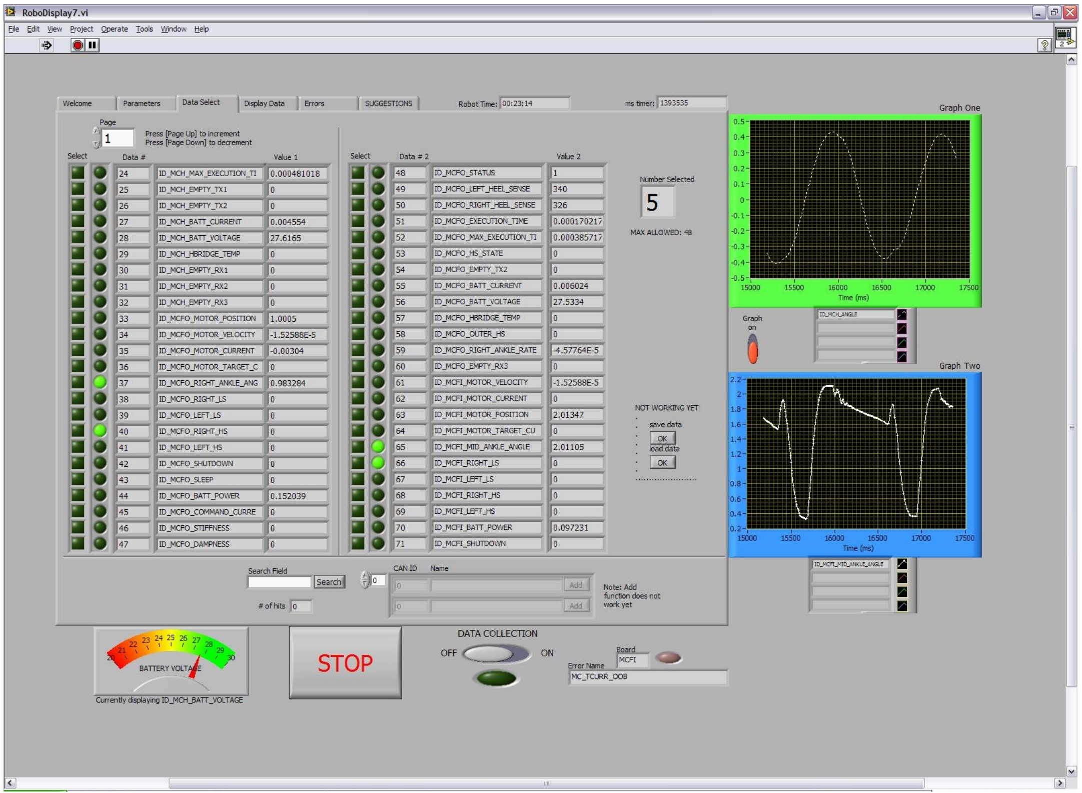
This LabVIEW-based software running on a Windows PC communicates data to and from the robot through a high-speed serial port, usually an external FTDI-brand USB to serial converter. From there the data either goes directly to a serial port on the back of Ranger by wire, or through a Bluetooth data link. LabVIEW was selected for ease of programming data acquisition features, and in particular for the ease with which multiple parallel threads of operation can be set up. In this case one thread parses and logs data streaming in from the serial port, another handles data transmission, and a third takes care of the user interface and graph display. The RoboDAQ program includes several useful features for studying the robot's performance:

First, it allows the user to look at graphs of data in real time, while simultaneously logging data to the hard disk. All of the variables in the system, possibly many hundreds, are received and logged periodically, but at a slow 1 Hz rate. However, up to 48 robot variables can be selected for real-time plotting and logging at higher data rates. These requests are relayed to a function on the LPC3250 which puts the desired values into a buffer for transmission to the laptop. By requesting a higher speed, variables can be graphed and logged to a file at up to about 250 Hz over Bluetooth and 500 Hz with the direct RS-232 cable. The actual speed varies depending upon the quality of the wireless link, and how many variables are chosen for higher-speed transmission.

Second, data can be logged to a file on disk for later analysis, generally by a custom MATLAB-based plotting program. One of these MATLAB programs allows synchronization of the data plots with robot video.

Third, users can adjust parameters on the robot while it is walking, using the Bluetooth link. RoboDAQ also allows saving sets of parameters for later use, and loading these saved parameters to the robot. For the walking distance record attempts, parameter adjustment was turned off, and Ranger relied on parameters compiled into its code. Only the data logging features were used.

Fourth, the LabVIEW program reports error conditions from all levels of the Ranger code. Error monitoring is critical, when working with a complex device like Ranger, to make certain that unexpected error conditions are not corrupting the results of the testing.


RoboDisplay7.vi
File Edit View Project Operate Tools Window Help
Welcome
Parameters
Data Select
Display Data
Errors
SUGGESTIONS
Graph One
Graph Two
Time (ms)
15000
15500
16000
16500
17000
17500
Graph on
ID_MCH_ANGLE
ID_MCFI_MID_ANKLE_ANGLE
Search
Add
OK
STOP
BATTERY VOLTAGE
OFF
ON

Robot Time: 00:23:14

ms timer: 1393535

Page: 1

Press [Page Up] to increment
Press [Page Down] to decrement

| Data # | | Value 1 |
|---|---|---|
| 24 | ID_MCH_MAX_EXECUTION_TI | 0.000481018 |
| 25 | ID_MCH_EMPTY_TX1 | 0 |
| 26 | ID_MCH_EMPTY_TX2 | 0 |
| 27 | ID_MCH_BATT_CURRENT | 0.004554 |
| 28 | ID_MCH_BATT_VOLTAGE | 27.6165 |
| 29 | ID_MCH_HBRIDGE_TEMP | 0 |
| 30 | ID_MCH_EMPTY_RX1 | 0 |
| 31 | ID_MCH_EMPTY_RX2 | 0 |
| 32 | ID_MCH_EMPTY_RX3 | 0 |
| 33 | ID_MCFO_MOTOR_POSITION | 1.0005 |
| 34 | ID_MCFO_MOTOR_VELOCITY | -1.52588E-5 |
| 35 | ID_MCFO_MOTOR_CURRENT | -0.00304 |
| 36 | ID_MCFO_MOTOR_TARGET_C | 0 |
| 37 | ID_MCFO_RIGHT_ANKLE_ANG | 0.983284 |
| 38 | ID_MCFO_RIGHT_LS | 0 |
| 39 | ID_MCFO_LEFT_LS | 0 |
| 40 | ID_MCFO_RIGHT_HS | 0 |
| 41 | ID_MCFO_LEFT_HS | 0 |
| 42 | ID_MCFO_SHUTDOWN | 0 |
| 43 | ID_MCFO_SLEEP | 0 |
| 44 | ID_MCFO_BATT_POWER | 0.152039 |
| 45 | ID_MCFO_COMMAND_CURRE | 0 |
| 46 | ID_MCFO_STIFFNESS | 0 |
| 47 | ID_MCFO_DAMPNESS | 0 |

| Data # 2 | | Value 2 |
|---|---|---|
| 48 | ID_MCFO_STATUS | 1 |
| 49 | ID_MCFO_LEFT_HEEL_SENSE | 340 |
| 50 | ID_MCFO_RIGHT_HEEL_SENSE | 326 |
| 51 | ID_MCFO_EXECUTION_TIME | 0.000170217 |
| 52 | ID_MCFO_MAX_EXECUTION_TI | 0.000385717 |
| 53 | ID_MCFO_HS_STATE | 0 |
| 54 | ID_MCFO_EMPTY_TX2 | 0 |
| 55 | ID_MCFO_BATT_CURRENT | 0.006024 |
| 56 | ID_MCFO_BATT_VOLTAGE | 27.5334 |
| 57 | ID_MCFO_HBRIDGE_TEMP | 0 |
| 58 | ID_MCFO_OUTER_HS | 0 |
| 59 | ID_MCFO_RIGHT_ANKLE_RATE | -4.57764E-5 |
| 60 | ID_MCFO_EMPTY_RX3 | 0 |
| 61 | ID_MCFI_MOTOR_VELOCITY | -1.52588E-5 |
| 62 | ID_MCFI_MOTOR_CURRENT | 0 |
| 63 | ID_MCFI_MOTOR_POSITION | 2.01347 |
| 64 | ID_MCFI_MOTOR_TARGET_CU | 0 |
| 65 | ID_MCFI_MID_ANKLE_ANGLE | 2.01105 |
| 66 | ID_MCFI_RIGHT_LS | 0 |
| 67 | ID_MCFI_LEFT_LS | 0 |
| 68 | ID_MCFI_RIGHT_HS | 0 |
| 69 | ID_MCFI_LEFT_HS | 0 |
| 70 | ID_MCFI_BATT_POWER | 0.097231 |
| 71 | ID_MCFI_SHUTDOWN | 0 |

Number Selected: 5

MAX ALLOWED: 48

NOT WORKING YET
save data
load data

Search Field

# of hits: 0

| CAN ID | Name |
|---|---|
| 0 | |
| 0 | |

Note: Add function does not work yet

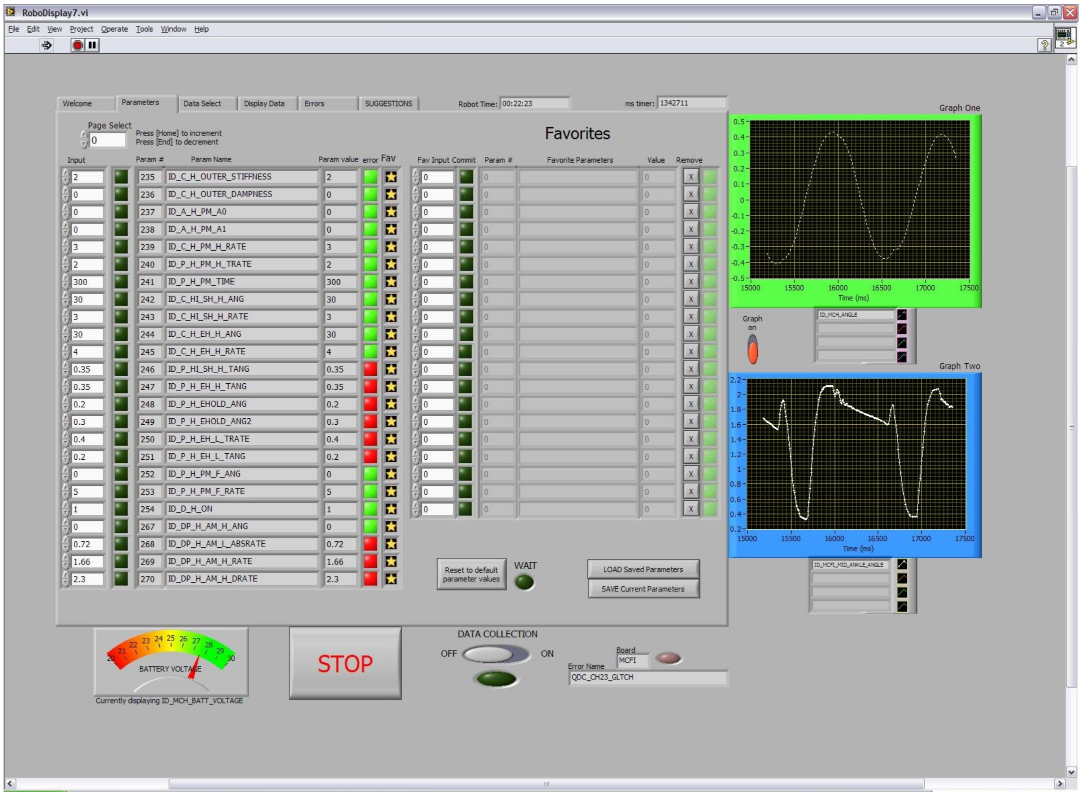
Currently displaying ID_MCH_BATT_VOLTAGE

DATA COLLECTION

Board: MCFI

Error Name: MC_TCURR_OOB

(a) Sensor Variables


RoboDisplay7.vi
File Edit View Project Operate Tools Window Help
Welcome
Parameters
Data Select
Display Data
Errors
SUGGESTIONS
Graph One
Graph Two
Time (ms)
15000
15500
16000
16500
17000
17500
Graph on
ID_MCH_ANGLE
ID_MCFI_MID_ANKLE_ANGLE
Reset to default parameter values
WAIT
LOAD Saved Parameters
SAVE Current Parameters
STOP
BATTERY VOLTAGE
OFF
ON

Robot Time: 00:22:23

ms timer: 1342711

Page Select: 0

Press [Home] to increment
Press [End] to decrement

| Input | Param # | Param Name | Param value | error |
|---|---|---|---|---|
| 2 | 235 | ID_C_H_OUTER_STIFFNESS | 2 | 🟩 |
| 0 | 236 | ID_C_H_OUTER_DAMPNESS | 0 | 🟩 |
| 0 | 237 | ID_A_H_PM_A0 | 0 | 🟩 |
| 0 | 238 | ID_A_H_PM_A1 | 0 | 🟩 |
| 3 | 239 | ID_C_H_PM_H_RATE | 3 | 🟩 |
| 2 | 240 | ID_P_H_PM_H_TRATE | 2 | 🟩 |
| 300 | 241 | ID_P_H_PM_TIME | 300 | 🟩 |
| 30 | 242 | ID_C_HI_SH_H_ANG | 30 | 🟩 |
| 3 | 243 | ID_C_HI_SH_H_RATE | 3 | 🟩 |
| 30 | 244 | ID_C_H_EH_H_ANG | 30 | 🟩 |
| 4 | 245 | ID_C_H_EH_H_RATE | 4 | 🟩 |
| 0.35 | 246 | ID_P_HI_SH_H_TANG | 0.35 | 🟥 |
| 0.35 | 247 | ID_P_H_EH_H_TANG | 0.35 | 🟥 |
| 0.2 | 248 | ID_P_H_EHOLD_ANG | 0.2 | 🟥 |
| 0.3 | 249 | ID_P_H_EHOLD_ANG2 | 0.3 | 🟥 |
| 0.4 | 250 | ID_P_H_EH_L_TRATE | 0.4 | 🟥 |
| 0.2 | 251 | ID_P_H_EH_L_TANG | 0.2 | 🟥 |
| 0 | 252 | ID_P_H_PM_F_ANG | 0 | 🟩 |
| 5 | 253 | ID_P_H_PM_F_RATE | 5 | 🟩 |
| 1 | 254 | ID_D_H_ON | 1 | 🟩 |
| 0 | 267 | ID_DP_H_AM_H_ANG | 0 | 🟩 |
| 0.72 | 268 | ID_DP_H_AM_L_ABSRATE | 0.72 | 🟥 |
| 1.66 | 269 | ID_DP_H_AM_H_RATE | 1.66 | 🟥 |
| 2.3 | 270 | ID_DP_H_AM_H_DRATE | 2.3 | 🟥 |

Favorites

| Fav Input | Param # | Favorite Parameters | Value |
|---|---|---|---|
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |
| 0 | 0 | | 0 |

Currently displaying ID_MCH_BATT_VOLTAGE

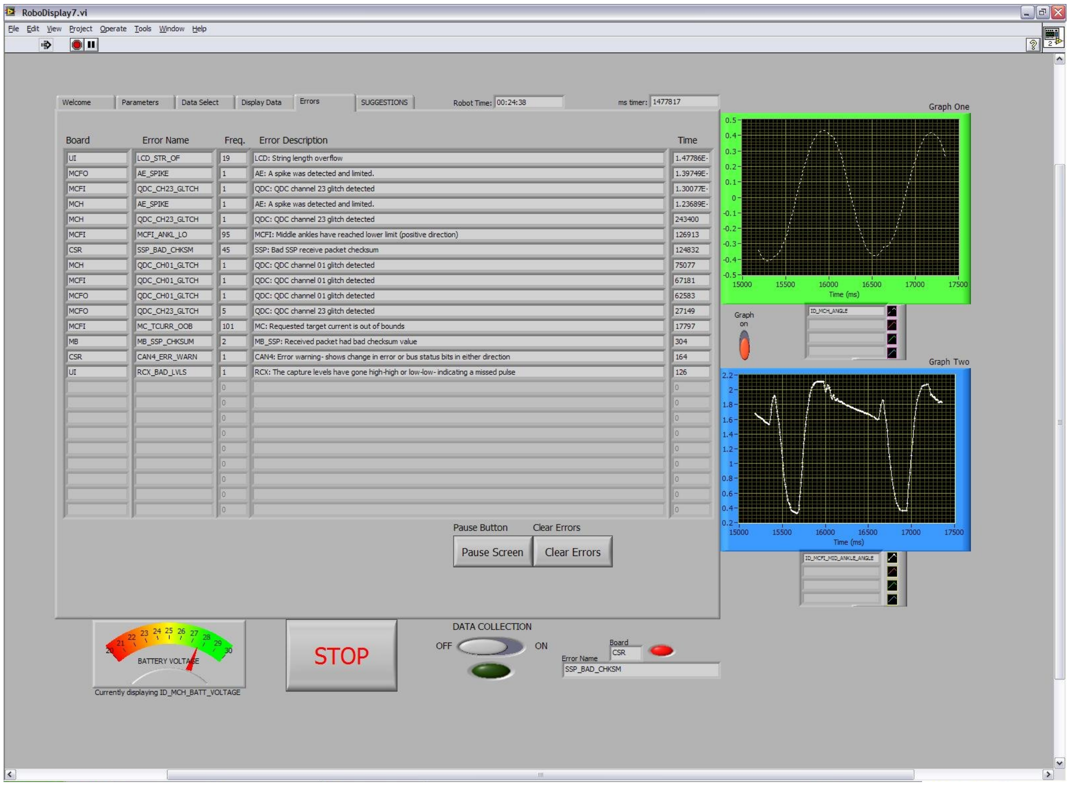
DATA COLLECTION

Board: MCFI

Error Name: QDC_CH23_GLTCH

(b) Parameter Variables

Figure 13: **Snapshots of LabVIEW based program on a Windows PC that communicates with the robot.** There is a tab for **(a)** display of sensor variables, and **(b)** display of parameter variables


RoboDisplay7.vi
File Edit View Project Operate Tools Window Help
Welcome
Parameters
Data Select
Display Data
Errors
SUGGESTIONS
Graph One
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
15000
15500
16000
16500
17000
17500
Time (ms)
Graph on
ID_MCH_ANGLE
Graph Two
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
15000
15500
16000
16500
17000
17500
Time (ms)
ID_MCFI_MID_ANKLE_ANGLE
Pause Button
Pause Screen
Clear Errors
Clear Errors
20 21 22 23 24 25 26 27 28 29 30
BATTERY VOLTAGE
STOP
DATA COLLECTION
OFF
ON

Robot Time: 00:24:38

ms timer: 1477817

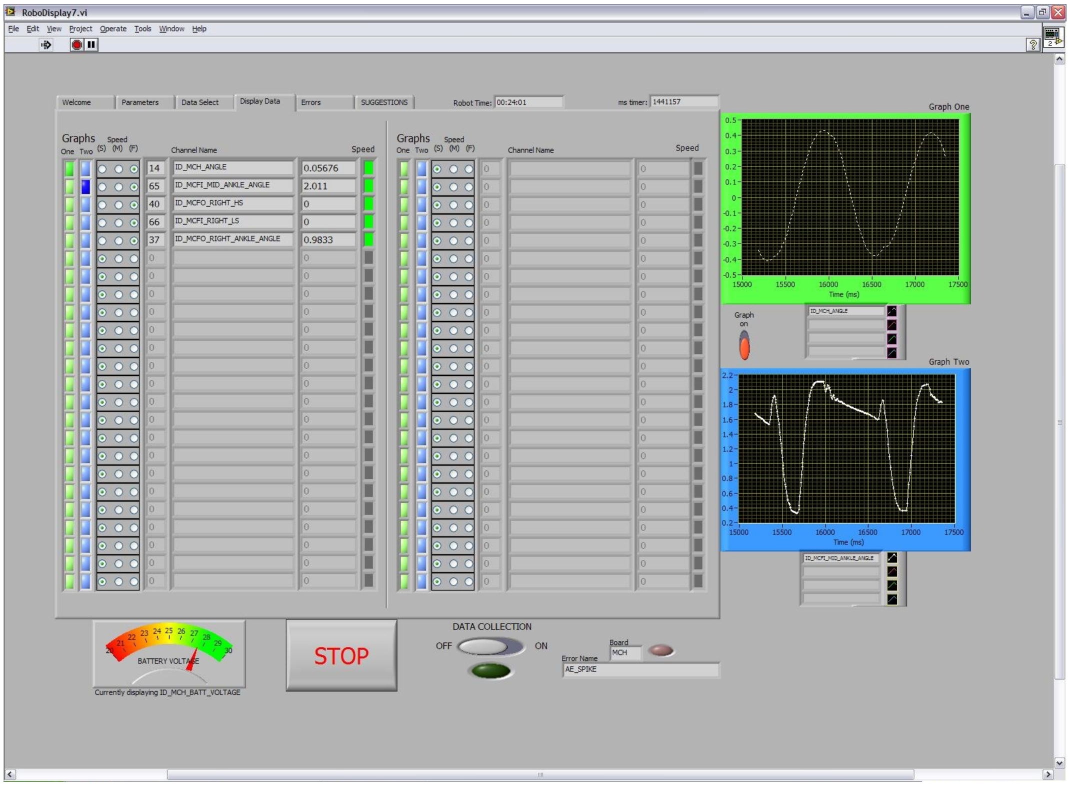
| Board | Error Name | Freq. | Error Description | Time |
|---|---|---|---|---|
| UI | LCD_STR_OF | 19 | LCD: String length overflow | 1.47786E- |
| MCFO | AE_SPIKE | 1 | AE: A spike was detected and limited. | 1.39749E- |
| MCFI | QDC_CH23_GLTCH | 1 | QDC: QDC channel 23 glitch detected | 1.30077E- |
| MCH | AE_SPIKE | 1 | AE: A spike was detected and limited. | 1.23689E- |
| MCH | QDC_CH23_GLTCH | 1 | QDC: QDC channel 23 glitch detected | 243400 |
| MCFI | MCFI_ANKL_LO | 95 | MCFI: Middle ankles have reached lower limit (positive direction) | 126913 |
| CSR | SSP_BAD_CHKSM | 45 | SSP: Bad SSP receive packet checksum | 124832 |
| MCH | QDC_CH01_GLTCH | 1 | QDC: QDC channel 01 glitch detected | 75077 |
| MCFI | QDC_CH01_GLTCH | 1 | QDC: QDC channel 01 glitch detected | 67181 |
| MCFO | QDC_CH01_GLTCH | 1 | QDC: QDC channel 01 glitch detected | 62583 |
| MCFO | QDC_CH23_GLTCH | 5 | QDC: QDC channel 23 glitch detected | 27149 |
| MCFI | MC_TCURR_OOB | 101 | MC: Requested target current is out of bounds | 17797 |
| MB | MB_SSP_CHKSUM | 2 | MB_SSP: Received packet had bad checksum value | 304 |
| CSR | CAN4_ERR_WARN | 1 | CAN4: Error warning- shows change in error or bus status bits in either direction | 164 |
| UI | RCX_BAD_LVLS | 1 | RCX: The capture levels have gone high-high or low-low- indicating a missed pulse | 126 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |
|  |  | 0 |  | 0 |

Currently displaying ID_MCH_BATT_VOLTAGE

Board: CSR

Error Name: SSP_BAD_CHKSM

(c) Errors


RoboDisplay7.vi
File Edit View Project Operate Tools Window Help
Welcome
Parameters
Data Select
Display Data
Errors
SUGGESTIONS
Graph One
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
15000
15500
16000
16500
17000
17500
Time (ms)
Graph on
ID_MCH_ANGLE
Graph Two
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
15000
15500
16000
16500
17000
17500
Time (ms)
ID_MCFI_MID_ANKLE_ANGLE
20 21 22 23 24 25 26 27 28 29 30
BATTERY VOLTAGE
STOP
DATA COLLECTION
OFF
ON

Robot Time: 00:24:01

ms timer: 1441157

| Speed (S) | Speed (M) | Speed (F) |  | Channel Name | Speed |
|---|---|---|---|---|---|
| ○ | ○ | ● | 14 | ID_MCH_ANGLE | 0.05676 |
| ○ | ○ | ● | 65 | ID_MCFI_MID_ANKLE_ANGLE | 2.011 |
| ○ | ○ | ● | 40 | ID_MCFO_RIGHT_HS | 0 |
| ○ | ○ | ● | 66 | ID_MCFI_RIGHT_LS | 0 |
| ○ | ○ | ● | 37 | ID_MCFO_RIGHT_ANKLE_ANGLE | 0.9833 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |

| Speed (S) | Speed (M) | Speed (F) |  | Channel Name | Speed |
|---|---|---|---|---|---|
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |
| ● | ○ | ○ | 0 |  | 0 |

Currently displaying ID_MCH_BATT_VOLTAGE

Board: MCH

Error Name: AE_SPIKE

(d) User selects sensor variables for monitoring and data collection.

Figure 14: **Snapshots of LabVIEW based program on a Windows PC that communicates with the robot.** There is a tab for **(c)** display of various errors on the robot and **(d)** tab to monitor and collect data from a user defined set from the sensor variables in (a).

**Bluetooth.** The Bluetooth data module is the RoboDAQ program's counterpart on the robot itself. Running on the top-level ARM9 processor, its job is to look at the data requests from RoboDAQ, updates to the central data structure, and the available transmission bandwidth, and then decide which and how many data points should be put into the serial port buffers and transmitted. For this purpose the code maintains a high-speed data list and a medium-speed data list; all other data is by default slow-speed. For each transmission cycle the Bluetooth module sends all the data in the high-speed list, one data point from the medium-speed list, and one from the slow-speed list. The relative speed of the lists is determined by the lengths of the lists. For example, medium-speed data only gets sent at a higher rate than the slow-speed data if its list is shorter; similarly, if the fast list and the medium list each have only one item, they will be sent at the same fast speed. The slow data is sent whether it has changed since the previous transmission or not; fast and medium-speed data is only transmitted when the values change, to save bandwidth. The transmission rate for a direct-wire serial connection is quite consistent, but the Bluetooth connection speed varies with signal quality, and periodically shuts down for short periods of time, presumably for internal protocol maintenance. These transmission gaps make it difficult to maintain a steady, uninterrupted transmission rate via Bluetooth. A proportional controller was added on the transmit rate, tuned to try to keep a large output buffer half-filled; this helped, but when the Bluetooth link drops out for too long the buffer eventually overflows and leaves a gap in the data log.

## 4.12 Other code modules

Ranger uses about 25 other code modules, ranging from a few small files to read in pushbutton switch values and light up displays and LEDs, to more complex parsing and estimating code for sensors like the IMU and foot contact sensors. See the code itself and associated documentation for more information (available online). c

# 5 Equations of motion

The robot model is described in the main paper and shown in figure 1b therein. The dimensions are shown in figure 1 below. Reference frames and angles are shown in figure 15.

**Coordinates.** During double stance we can write the co-ordinates of the hip $x_h, y_h$, using the fixed (Newtonian) coordinate system $xy$ in two ways (see figure 15): 1) using the path $\mathrm{OP_1A_1H}$, and 2) using the path $\mathrm{OP_2A_2H}$. We thus have two kinematic restrictions (constraints) on the joint angles ($q_2, q_3$ and $q_4$), the absolute angle of the lead foot $q_1$ and step length ($x_{p2} - x_{p1}$) during double stance.

$$\begin{aligned} x_h &= x_h \\ \implies x_{p1} + l\sin(q_1+q_2) - d\sin(q_1) - rq_1 &= x_{p2} - l\sin(q_3-q_1-q_2) - d\sin(q_1+q_2-q_3-q_4)\ldots \\ &\quad - r(q_1+q_2-q_3-q_4) \end{aligned} \tag{4}$$

$$\begin{aligned} y_h &= y_h \\ \implies r - l\cos(q_1+q_2) + d\cos(q_1) &= r - l\cos(q_3-q_1-q_2) + d\cos(q_1+q_2-q_3-q_4) \end{aligned} \tag{5}$$

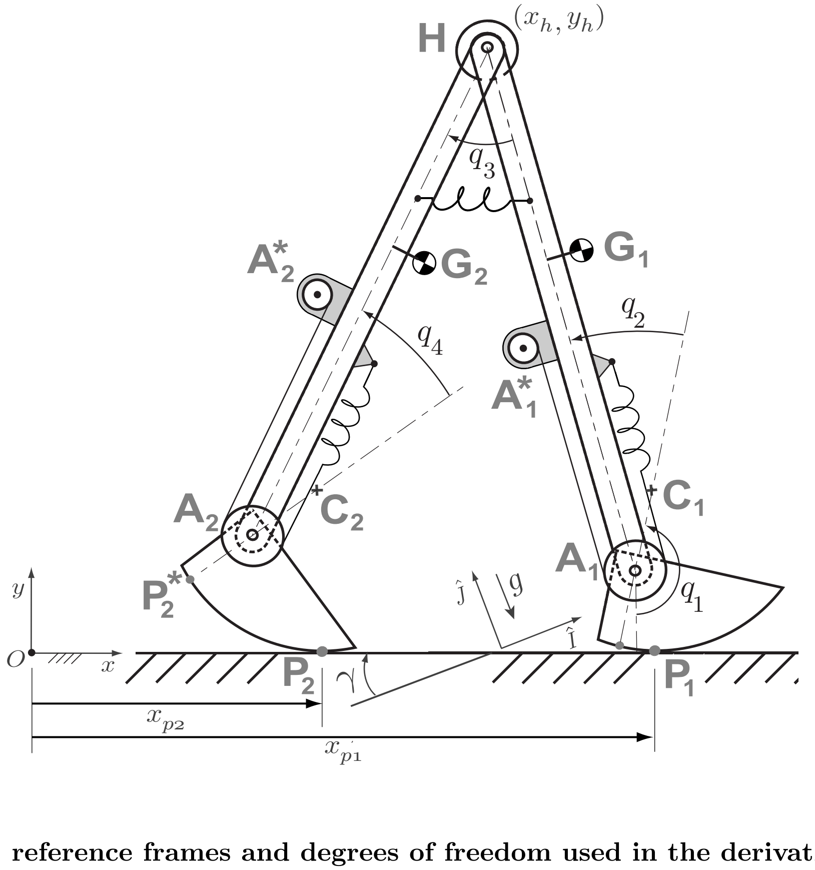


Figure 15: **Robot reference frames and degrees of freedom used in the derivation of the equations of motion.** The absolute angle made by the lead foot on the ground with the vertical is $q_1$. Joint angles are $q_2$, $q_3$ and $q_4$. Hip motor angle is the same as hip joint angle $q_3$. Ankle motor angles associated with the joint $A_1^\star$ is $q_{2m}$ and with joint $A_2^\star$ is $q_{4m}$.

## 5.1 Equations of motion during single and double stance

The governing differential equations are found using the free body diagrams (FBDs) shown in figures 16 and 17, respectively, and using angular momentum balance about judiciously chosen points that eliminate (at least some of) the constraint forces.

**Double stance.** In double stance the robot has 2 kinematic degrees of freedom and two motor degrees of freedom. Releasing one foot from the ground and then constraining it, we think of this rather as 4 kinematic degrees of freedom, 2 motor degrees of freedom and 2 kinematic constraint equations. From figure 16 we use the systems defined by the free body diagrams (a) – (d). We then use angular momentum balance about the points $P_1, A_1, H$ and $A_2$, respectively, to generate

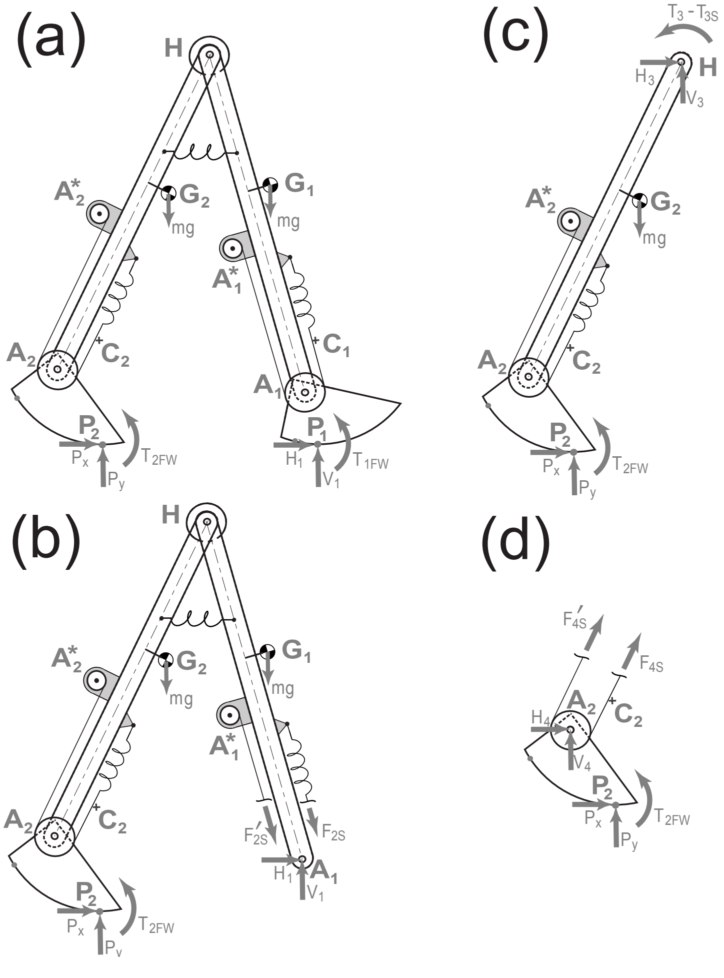

(a)
H
$A_2^*$
$G_2$
mg
$G_1$
mg
$A_1^*$
$C_2$
$C_1$
$A_2$
$A_1$
$P_2$
$P_x$
$P_y$
$T_{2FW}$
$P_1$
$H_1$
$V_1$
$T_{1FW}$
(b)
H
$A_2^*$
$G_2$
mg
$G_1$
mg
$A_1^*$
$C_2$
$A_2$
$F_{2S}'$
$F_{2S}$
$H_1$
$A_1$
$V_1$
$P_2$
$P_x$
$P_y$
$T_{2FW}$
(c)
$T_3 - T_{3S}$
$H_3$
H
$V_3$
$A_2^*$
$G_2$
mg
$A_2$
$C_2$
$P_2$
$P_x$
$P_y$
$T_{2FW}$
(d)
$F_{4S}'$
$F_{4S}$
$H_4$
$A_2$
$C_2$
$V_4$
$P_2$
$P_x$
$P_y$
$T_{2FW}$

Figure 16: **Free Body Diagrams (FBD) to derive equations for double stance.** We have four free body diagrams. The arrows indicate all of the non-neglected forces and torques acting on each of the four systems.

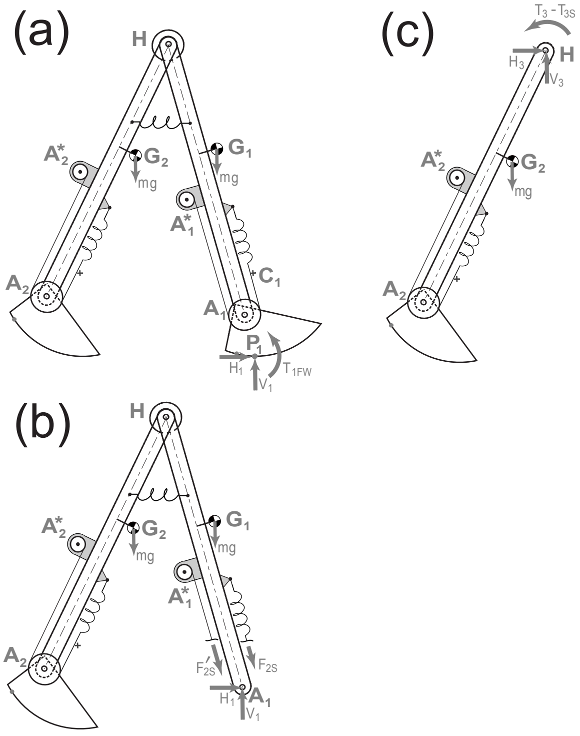


Figure 17: **Free Body Diagrams (FBD) to derive equation for single stance.** We have 3 free body diagrams. Because the feet are massless there is no information in drawing a FBD of the swing foot.

4 equations. The motor degrees of freedom are described with the motor equations from the main paper (equations 7 and 8 below). Two additional constraint equations for double stance are obtained by taking the second derivatives of equations 4 and 5,

$$\dot{\overrightarrow{H}}_{/i} = \overrightarrow{M}_{/i} \qquad \text{where } i = P_1, A_1, H, A_2 \tag{6}$$

$$T_{2S} = G_A(KI_2 - G_A J_m \ddot{q}_{2m}) - T_{fA}(I_2, \dot{q}_{2m}) \tag{7}$$

$$T_{4S} = G_A(KI_4 - G_A J_m \ddot{q}_{4m}) - T_{fA}(I_4, \dot{q}_{4m}) \tag{8}$$

$$\ddot{x}_h = \ddot{x}_h \tag{9}$$

$$\ddot{y}_h = \ddot{y}_h \tag{10}$$

$\overrightarrow{M}_{/i}$ and $\dot{\overrightarrow{H}}_{/i}$ are the sum of external torques and rate of change of angular momentum about the point $i$. These expressions are shown expanded in section 5.3. Note, from the main text the spring torque at joint 2 is, $T_{2S} = F_{2S} r_p = k r_p^2 (q_{2m} - q_2) = k_s (q_{2m} - q_2)$ and spring torque at joint 4 is $T_{4S} = F_{4S} r_p = k r_p^2 (q_{4m} - q_4) = k_s (q_{4m} - q_4)$. $F_{2S}$ and $F_{4S}$ are the spring forces in the springs at joint 2 and 4 respectively, $r_p$ is the radius of the ankle pulley and $k$ is the linear spring constant of the cable joining ankle to the ankle motor. Please see the nomenclature (section 1) for definitions of terms.

Altogether we have 6 differential equations of motion and two differential equations from differentiating the closed-linkage geometric constraint. From these we can solve, at any instant in time, for the angular accelerations of each robot part, both ankle motors, and the ground contact force at one foot.

**Single stance.** The swing-foot is airborne for the single stance phase. Because we neglect the masses of the feet, the swing foot and swing motor have the same motions and the swing foot does not have independent motion. This eliminates one degree of freedom from the single stance phase. Thus, in the single stance phase we have five degrees of freedom (stance ankle angle, hip angle, two motor angles and the stance foot angle). Altogether we then have three kinematic degrees of freedom, and two motor degrees of freedom.

Using figure 17 we use the systems shown in the free body diagrams (a)–(c). With these we use angular momentum balance about the points $P_1, H$ and $A_1$ respectively, to generate 3 equations of motion. Two more equations come from the two ankle motor equations. Thus, we have the following equations

$$\begin{aligned} \dot{\overrightarrow{H}}_{/i} &= \overrightarrow{M}_{/i} && \text{where } i = P_1, A_1, H && (11)\\ T_{2S} &= G_A(KI_2 - G_A J_m \ddot{q}_{2m}) - T_{fA}(I_2, \dot{q}_{2m}) && && (12)\\ 0 &= G_A(KI_4 - G_A J_m \ddot{q}_{4m}) - T_{fA}(I_4, \dot{q}_{4m}) && && (13) \end{aligned}$$

$\overrightarrow{M}_{/i}$ and $\dot{\overrightarrow{H}}_{/i}$ are the sums of the external torques and the rate of change of angular momentum about the point $i$. In section 5.3, we give the detailed expansions of these expressions. The equations above make up 5 differential equations for the angles of three body parts (all but the swing foot) and the two motors.

**Matrix form of the governing equations.** Equations 6 to 10 can be re-arranged to get the following equation for double stance,

$$\mathbf{A}_{ds}\mathbf{X}_{ds} = \mathbf{b}_{ds} \tag{14}$$

where the unknown is the $8 \times 1$ vector, $\mathbf{X}_{ds} = [\ddot{q}_1 \ , \ \ddot{q}_2 \ , \ \ddot{q}_{2m} \ , \ \ddot{q}_3 \ , \ \ddot{q}_4 \ , \ \ddot{q}_{4m} \ , \ P_x \ , \ P_y]'$. At a given dynamical state (given angles and rates) the $8 \times 8$ matrix $\mathbf{A}_{ds}$ and the $8 \times 1$ vector $\mathbf{b}_{ds}$ are known.

**Extracting the elements of $\mathbf{A}_{ds}$ and $\mathbf{b}_{ds}$.** There are various ways to find the elements of $\mathbf{A}_{ds}$ and $\mathbf{b}_{ds}$. Here is our somewhat clumsy method. We use symbolic algebra to evaluate the eight equations given in equation 6 to equation 10. Our next goal is to compute symbolic values of the individual elements of the matrices in $\mathbf{A}_{ds}$ and vector $\mathbf{b}_{ds}$. The first element of $\mathbf{b}_{ds}$ i.e. $b_1$ is obtained by setting $\mathbf{X}_{ds} = [0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0]$ and evaluating the first equation in 6. Similarly, we can calculate the other elements of $\mathbf{b}_{ds}$. To get the first row and first column element of $\mathbf{A}_{ds}$ i.e. $A_{11}$, we evaluate first equation in 6 by putting $\mathbf{X}_{ds} = [1 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0 \ , \ 0]$ and from this value subtract out $b_1$. Following a similar procedure it is possible to get every element of the matrix $\mathbf{A}_{ds}$. As pointed out by Manoj Srinivasan (private communication) the MATLAB symbolic command JACOBIAN could have simplified this extraction.

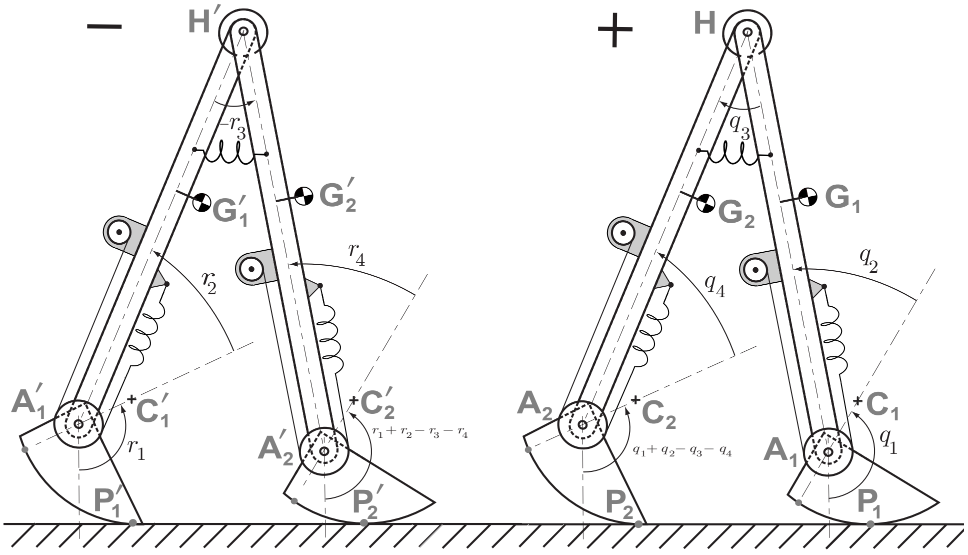


Figure 18: **Angle swap for heel-strike derivations.** The instant just before heel-strike is denoted by $^-$ and the instant just after heel-strike by $^+$. We swap the names of the legs during heel-strike as shown. To simplify notation the angles are named $r_i$ before collision and $q_i$ after collision., where $i$ is the joint number.

**Single stance equations in matrix form.** Equations 11 to 13 can be re-arranged to get the following equation for single stance.

$$\mathbf{A}_{ss}\mathbf{X}_{ss} = \mathbf{b}_{ss} \tag{15}$$

where the unknown is the $6 \times 1$ vector, $\mathbf{X}_{ss} = [\ddot{q}_1\ ,\ \ddot{q}_2\ ,\ \ddot{q}_{2m}\ ,\ \ddot{q}_3\ ,\ \ddot{q}_4\ ,\ \ddot{q}_{4m}\ ]'$, while the $6 \times 6$ matrix $\mathbf{A}_{ss}$ and the $6 \times 1$ vector $\mathbf{b}_{ss}$ are known and can be found in a similar way as found for the double stance equations.

## 5.2 Collisional heel-strike equations

Here we consider the jump (the discontinuity) in angular rates when the swing foot collides with the ground at heel-strike. We consider first the case when this is a transition from a single-stance phase to a double-stance phase.

Figure 18 shows the robot an instant before heel-strike, denoted by $^-$ and an instant after heel-strike, denoted by $^+$. We are interested in finding the angles after heelstrike, i.e.
$[\mathbf{q}] = [q_1\ ,\ q_2\ ,\ q_{2m}\ ,\ q_3\ ,\ q_4\ ,\ q_{4m}]'$ and angular velocities after heel-strike, i.e.
$[\dot{\mathbf{q}}^+] = [\dot{q}_1\ ,\ \dot{q}_2\ ,\ \dot{q}_{2m}\ ,\ \dot{q}_3\ ,\ \dot{q}_4\ ,\ \dot{q}_{4m}]'$.

The angles after heel-strike are found by using figure 18 and swapping the angles to generate equation 16. The velocities of joints after heel-strike are found by conservation of angular momentum. Doing conservation of angular momentum about appropriate points as shown in figure 19 (a)–(d) we get equations 17. We assume that the motors (buffered by the ankle spring) do not participate in the heel-strike. We swap the motor velocities to get equations 18. Finally, two additional constraint equations are generated by taking the first derivatives of equations 4 and 5

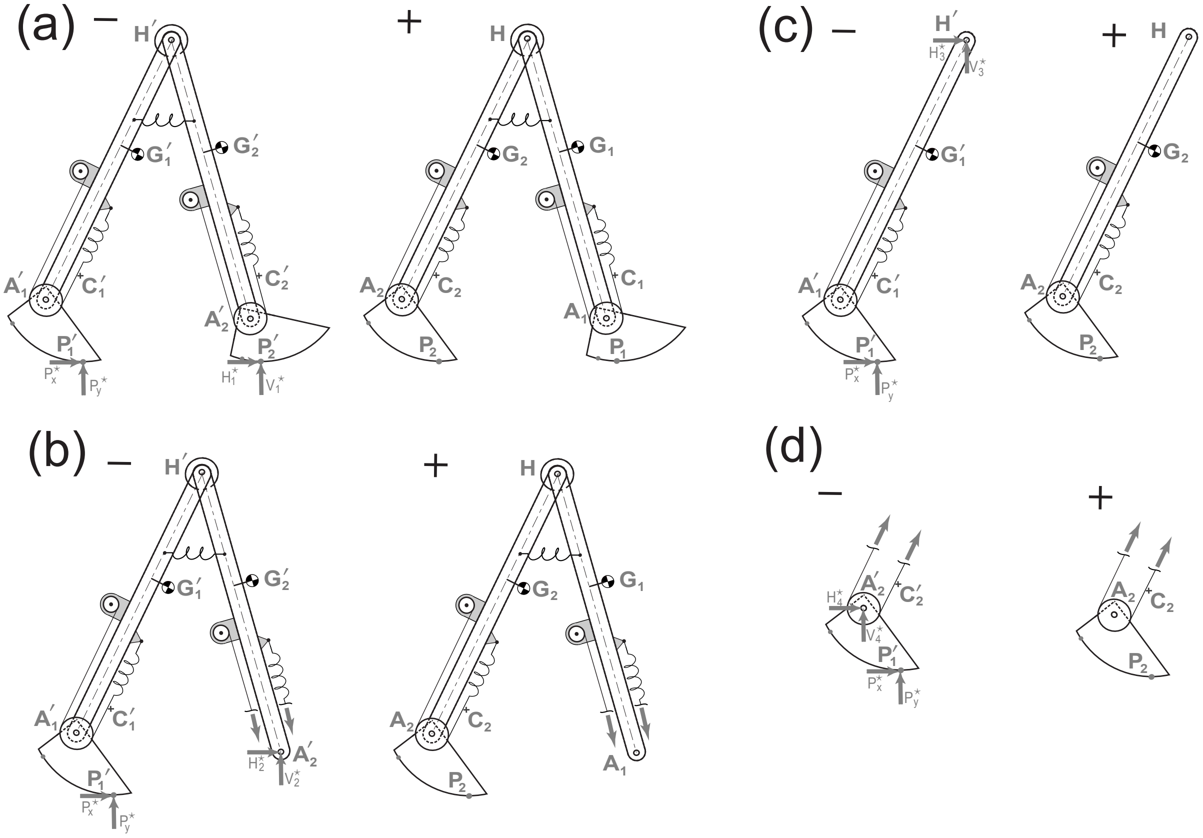

(a)
−
H′
G′1
G′2
A′1
C′1
P′1
Px*
Py*
C′2
A′2
P′2
H1*
V1*
+
H
G2
G1
A2
C2
C1
A1
P2
P1
(c)
−
H′
H3*
V3*
G′1
A′1
C′1
P′1
Px*
Py*
+
H
G2
A2
C2
P2
(b)
−
H′
G′1
G′2
A′1
C′1
P′1
Px*
Py*
A′2
H2*
V2*
+
H
G2
G1
A2
C2
P2
A1
(d)
−
H4*
A′2
C′2
V4*
P′1
Px*
Py*
+
A2
C2
P2

Figure 19: **Free Body Diagrams (FBDs) for heel-strike discontinuity.** The case shown is for the transition to a double-stance phase; there are impulses at both feet. The instant just before and after heel-strike is indicated by $^-$ and $^+$ respectively. The collisional forces are shown in the $^-$ configuration. We have four subsystems corresponding to the four body parts. The ankle motors are buffered by the ankle springs and do not participate in the collision; so the inner and outer ankle motors and rates simply exchange values (ie., keep their values and exchange their names).

to give equation 19.

$$\begin{aligned} q_1 &= r_1 + r_2 - r_3 - r_4 & q_2 &= r_4 \\ q_{2m} &= r_{4m} & q_3 &= -r_3 \\ q_4 &= r_2 & q_{4m} &= r_{2m} \end{aligned} \tag{16}$$

$$\begin{aligned} \overrightarrow{H}^{+}_{/P_1} &= \overrightarrow{H}^{-}_{/P_2'} + \vec{r}_{P_1'/P_2'} \times \vec{P}^{\star} & \overrightarrow{H}^{+}_{/A_1} &= \overrightarrow{H}^{-}_{/A_2'} + \vec{r}_{P_1'/A_2'} \times \vec{P}^{\star} \\ \overrightarrow{H}^{+}_{/H} &= \overrightarrow{H}^{-}_{/H'} + \vec{r}_{P_1'/H} \times \vec{P}^{\star} & \overrightarrow{H}^{+}_{/A_2} &= \overrightarrow{H}^{-}_{/A_1'} + \vec{r}_{P_1'/A_1'} \times \vec{P}^{\star} \end{aligned} \tag{17}$$

$$\dot{q}_{2m} = \dot{r}_{4m} \qquad \dot{q}_{4m} = \dot{r}_{2m} \tag{18}$$

$$\dot{x}_h = \dot{x}_h \qquad \dot{y}_h = \dot{y}_h \tag{19}$$

where $\overrightarrow{H}^{-}_{/i}$ and $\overrightarrow{H}^{+}_{/i}$ are the angular momentum about the point $i$ before and after heel-strke. In section 5.3, we give the detailed expansions of these expressions.

The equations 17 to 19 can be re-arranged to give the following equation for heel-strike phase,

$$\mathbf{A}_{hs}\mathbf{X}_{hs} = \mathbf{b}_{hs} \tag{20}$$

where the unknown is the $8 \times 1$ vector, $\mathbf{X}_{hs} = [\dot{q}_1 \ , \ \dot{q}_2 \ , \ \dot{q}_{2m} \ , \ \dot{q}_3 \ , \ \dot{q}_4 \ , \ \dot{q}_{4m} \ , \ P_x^\star \ , \ P_y^\star]'$, while the $8 \times 8$ matrix $\mathbf{A}_{hs}$ and the $8 \times 1$ vector $\mathbf{b}_{hs}$ are known and can be found in a similar way as found for the double stance equations.

**Single-stance to single-stance collisional transition.** The equations presented next are for gait sequences that do not have any double stance; i.e., the gait sequence is single stance → heel-strike → single stance. This sequence is not used on ranger, but is used as a special case to benchmark the equations of motion (section 6) and optimal trajectory (section 10).

The collisional jump equations for the single stance to single stance transition can be derived by similar means as used for the single stance to double stance transition as described before. We use the same figure 18 for this derivation, but 1) leave off the kinematic constraint that the former stance foot maintain contact; 2) leave off the related constraint impulses (set the impulses on the former stance foot to zero). The jump equations are:

$$q_1 = r_1 + r_2 - r_3 - r_4 \qquad q_3 = -r_3 \tag{21}$$

$$q_{2m} = r_{4m} \qquad q_2 = r_4 \tag{22}$$

$$q_4 = r_2 \qquad q_{4m} = r_{2m} \tag{23}$$

$$\overrightarrow{H}^{+}_{/P_1} = \overrightarrow{H}^{-}_{/P_2'} \tag{24}$$

$$\overrightarrow{H}^{+}_{/A_1} = \overrightarrow{H}^{-}_{/A_2'} \tag{25}$$

$$\overrightarrow{H}^{+}_{/H} = \overrightarrow{H}^{-}_{/H'} \tag{26}$$

$$\dot{q}_4 = \dot{r}_{2m} \qquad \dot{q}_{2m} = \dot{r}_{4m} \qquad \dot{q}_{4m} = \dot{r}_{2m} \tag{27}$$

$\overrightarrow{H}^{-}_{/i}$ and $\overrightarrow{H}^{+}_{/i}$ are the angular momentum about the point $i$ before and after heel-strke. Formulas for these follow.

## 5.3 Full expansion of terms in the single stance, double stance and heel-strike equations

The external moment $\overrightarrow{M}_{/i}$ and rate of change of angular momentum $\dot{\overrightarrow{H}}_{/i}$ about different points in the above equations are given next. We define $\delta_{DS} = 1$ in double stance and $\delta_{DS} = 0$ in single stance in the equations below.

$$\overrightarrow{M}_{/P_1} = \vec{r}_{G_1/P_1} \times m \, \vec{g} + \vec{r}_{G_2/P_1} \times m \, \vec{g} + \vec{r}_{P_2/P_1} \times \delta_{DS} \, \vec{P} + (T_{1FW} + \delta_{DS} T_{2FW})\hat{k}$$

$$
\begin{aligned}
&\dot{\overrightarrow{H}}_{/P_1} = \vec{r}_{G_1/P_1} \times m\, \vec{a}_{G_1} + J_\ell\, \vec{\alpha}_1 + \vec{r}_{G_2/P_1} \times m\, \vec{a}_{G_2} + J_\ell\, \vec{\alpha}_2 \\
&\overrightarrow{M}_{/A_1} = \vec{r}_{G_1/A_1} \times m\, \vec{g} + \vec{r}_{G_2/A_1} \times m\, \vec{g} + \vec{r}_{P_2/A_1} \times \delta_{DS}\, \vec{P} + (T_{2S} + \delta_{DS} T_{2FW})\hat{k} \\
&\dot{\overrightarrow{H}}_{/A_1} = \vec{r}_{G_1/A_1} \times m\, \vec{a}_{G_1} + J_\ell\, \vec{\alpha}_1 + \vec{r}_{G_2/A_1} \times m\, \vec{a}_{G_2} + J_\ell\, \vec{\alpha}_2 \\
&\overrightarrow{M}_{/H} = \vec{r}_{G_2/H} \times m\, \vec{g} + \vec{r}_{P_2/H} \times \delta_{DS}\, \vec{P} + (T_3 - T_{3S} + \delta_{DS} T_{2FW})\hat{k} \\
&\dot{\overrightarrow{H}}_{/H} = \vec{r}_{G_2/H} \times m\, \vec{a}_{G_2} + J_\ell\, \vec{\alpha}_2 \\
&\overrightarrow{M}_{/A_2} = \vec{r}_{P_2/A_2} \times \vec{P} + (T_{4S} + T_{2FW})\hat{k} && \text{Only in double stance} \\
&\dot{\overrightarrow{H}}_{/A_2} = 0 && \text{Only in double stance}
\end{aligned}
$$

The angular momentum $\overrightarrow{H}_{/i}$ about different points in the heel-strike equations are given next.

$$
\begin{aligned}
&\overrightarrow{H}^{+}_{/P_1} = \vec{r}_{G_1/P_1} \times m\, \vec{v}_{G_1} + J_\ell\, \vec{\omega}_1 + \vec{r}_{G_2/P_1} \times m\, \vec{v}_{G_2} + J_\ell\, \vec{\omega}_2 \\
&\overrightarrow{H}^{-}_{/P_2'} = \vec{r}_{G_1'/P_2'} \times m\, \vec{v}_{G_2'} + J_\ell\, \vec{\omega'}_1 + \vec{r}_{G_2'/P_2'} \times m\, \vec{v}_{G_2'} + J_\ell\, \vec{\omega'}_2 \\
&\overrightarrow{H}^{+}_{/A_1} = \vec{r}_{G_1/A_1} \times m\, \vec{v}_{A_1} + J_\ell\, \vec{\omega}_1 + \vec{r}_{G_2/A_1} \times m\, \vec{v}_{G_2} + J_\ell\, \vec{\omega}_2 \\
&\overrightarrow{H}^{-}_{/A_2'} = \vec{r}_{G_1'/A_2'} \times m\, \vec{v}_{G_2'} + J_\ell\, \vec{\omega'}_1 + \vec{r}_{G_2'/A_2'} \times m\, \vec{v}_{G_2'} + J_\ell\, \vec{\omega'}_2 \\
&\overrightarrow{H}^{+}_{/H} = \vec{r}_{G_2/H} \times m\, \vec{v}_{G_2} + J_\ell\, \vec{\omega}_2 \\
&\overrightarrow{H}^{-}_{/H'} = \vec{r}_{G_1'/H'} \times m\, \vec{v}_{G_1'} + J_\ell\, \vec{\omega'}_1 \\
&\overrightarrow{H}^{+}_{/A_2} = \overrightarrow{H}^{-}_{/A_1'} = \vec{0}
\end{aligned}
$$

## 6 Benchmark tests of the equations of motion

To validate the equations of motion we consider various special cases about which much is known. In particular, we reduced the equations of motion of Ranger to that of a passive walker, introduced a ramp, found stable limit cycles and compared our results with previously published results. Two special passive cases were considered: 1) a rimless wheel [1, 8], and the simplest walker [4, 5]. Without such checks how are we to trust our equations?

### 6.1 Recipe for analyzing passive dynamic walkers

A recipe for analyzing passive dynamic walkers has been presented in detail in Garcia's PhD thesis [5] (see Chapter 2) and in Coleman's PhD thesis [1] (see Chapter 1). Here is a summary.

1. Create a complete mechanical model of the walker. This includes defining model assumptions, defining model parameters, and deriving the equations of motion of the walker. The equations of motion generally involve smooth portions of the walk like single stance phase and discontinuous portions like heel-strike.

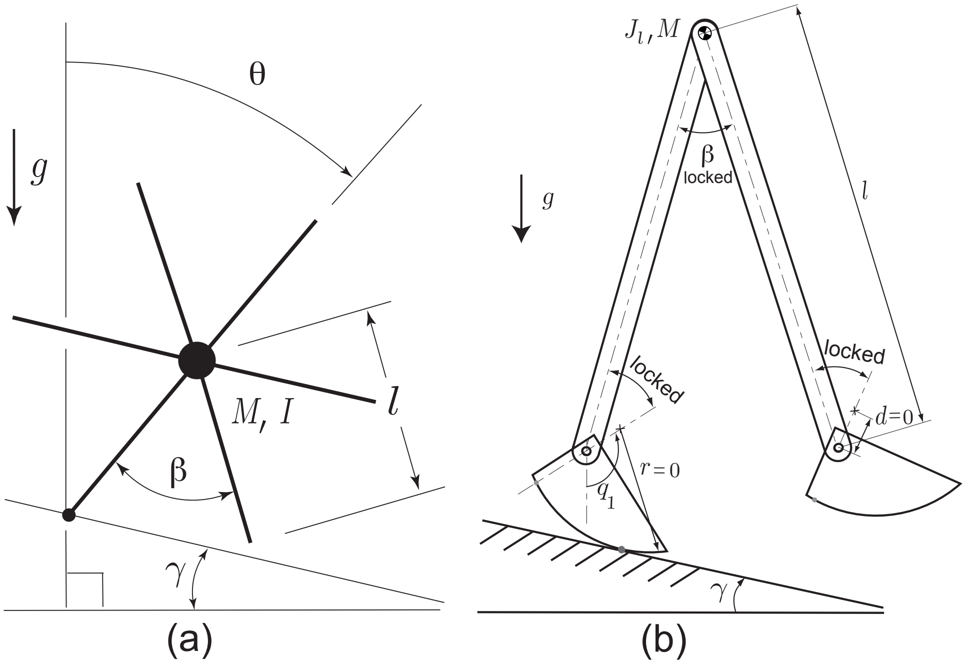


Figure 20: **2-D rimless wheel. (a)** 2-D rimless wheel analyzed by Coleman. (figure source: Coleman's PhD thesis [1]), **(b)** Ranger model simplified to a rimless wheel; The hip angle and ankle angles are locked and the centers of mass of the leg are put at the hip.

2. Define a Poincare map (McGeer's stride function) that maps the state of the system from one step to the next. Here we used the map $f$ from the state of the system $q_n$ just after one heel-strike to the state $q_{n+1}$ just after the next heel-strike. We use numerical root finding (e.g. Newton-Raphson method or bisection method) to find fixed points (roots of the function $f(q) - q$).
3. Find the Jacobian of the stride function at this root using numerical differentiation. If the magnitude of the biggest eigenvalue of the Jacobian is smaller than one than the system is stable, otherwise it is not.

## 6.2 Reduction to a 2-D rimless wheel

Figure 20 (a) shows the 2-D rimless wheel analyzed by Coleman. The mass at the center is $M$, legs have inertia $I_\ell$, legs length is $\ell$ and number of spokes is $n$. The inter-spoke angle $\beta$ is calculated from the number of spokes and is given by $\beta = 2\pi/n$. The ramp slope is $\gamma$. Coleman found that the roots and Jacobian of the stride function depend on the number of spokes $n$, the slope $\gamma$ and the non-dimensional term $\lambda^2 = M\ell^2/(I_\ell + M\ell^2)$. Coleman reports analytical results for $n = 6$, $\gamma = 0.2$ and $\lambda^2 = 2/3$ [2]. We will use these parameters for the benchmark.

**Ranger model reduced to a rimless wheel.** To derive the equations of motion we proceed as follows. First, we assume the gait sequence; single stance phase followed by heel-strike phase followed by single stance phase and so on. Next, we draw the free body diagram for Ranger model

(a) Rimless wheel

| Parameter | Value |
|---|---|
| $\ell$ | 1 m |
| $r$ | 0 |
| $d$ | 0 |
| $w$ | 0 |
| $c$ | 0 |
| $k_h$ | Not in equations |
| $k_s$ | Not in equations |
| $J_\ell$ | $0.25\ \mathrm{kg\,m^2}$ |
| $m$ | 0 |
| $M$ | 1 kg |
| $g$ | $1\ \mathrm{m/s^2}$ |
| $\gamma$ | 0.2 |
| $C_{1FW}$ | 0 |
| $C_{2FW}$ | Not in equations |

(b) Simplest walker

| Parameter | Value |
|---|---|
| $\ell$ | 1 m |
| $r$ | 0 |
| $d$ | 0 |
| $w$ | 0 |
| $c$ | 1 m |
| $k_h$ | Not in equations |
| $k_s$ | Not in equations |
| $J_\ell$ | 0 |
| $m$ | 1 kg |
| $M$ | $10^6$ kg |
| $g$ | $1\ \mathrm{m/s^2}$ |
| $\gamma$ | 0.009 |
| $C_{1FW}$ | 0 |
| $C_{2FW}$ | Not in equations |

Table 1: **Reduction of Ranger to simpler cases. (a)** Values of Ranger parameters for model reduction to a 2-D rimless wheel. **(b)** Ranger parameters for model reduction to simplest walker.

shown in figure 20 b). Finally, we use angular momentum balance about the foot contact points to derive the equations of motion.

Alternately, the equations of motion can be obtained from Ranger's equations presented earlier. We ignore the equation for angular momentum balance of the swing leg about the hip. And in all other equations we set the hip angle to be constant (no acceleration, no discontinuity in velocity at the collision). The equation of single stance are obtained from equation 11 with $i = P_1$. The equation for heel-strike is obtained from equations 21 and 24.

$$\begin{array}{lll}
\text{Single stance (continuous):} & \dot{\overrightarrow{H}}_{/P_1} = \overrightarrow{M}_{/P_1} & \\
\text{Heel-strike (instantaneous):} & q_1 = r_1 - r_3 & q_3 = -r_3 \\
 & \overrightarrow{H}^{+}_{/P_1} = \overrightarrow{H}^{-}_{/P_2'} &
\end{array}$$

Table 1 gives the parameters of the 2-D rimless wheel Ranger model.

**Comparison of fixed points.** We analyzed the simplified Ranger model with the parameters given in table 1 and using the recipe presented in section 6.1.

The fixed points based on analytical solutions is [2],
$(\theta_1^*,\dot{\theta}_1^*) = (\pi/n,\ -0.4603411266094583)$. Using an adaptive step integrator (Runge-Kutta 45) with integration tolerance of $10^{-13}$ and a similar root finder accuracy, we calculated the fixed points as $(q_1^*,\dot{q}_1^*) = (0.523598775598278,\ -0.460341126609482)$ which is accurate to the 13th decimal place.

**Eigenvalues.** The biggest eigenvalue based on analytical solution [2] is $\sigma = 4/9 = 0.4444444444444\bar{4}$. Using fixed point found earlier and using perturbation of $10^{-5}$, we calculated the Jacobian of the lin-

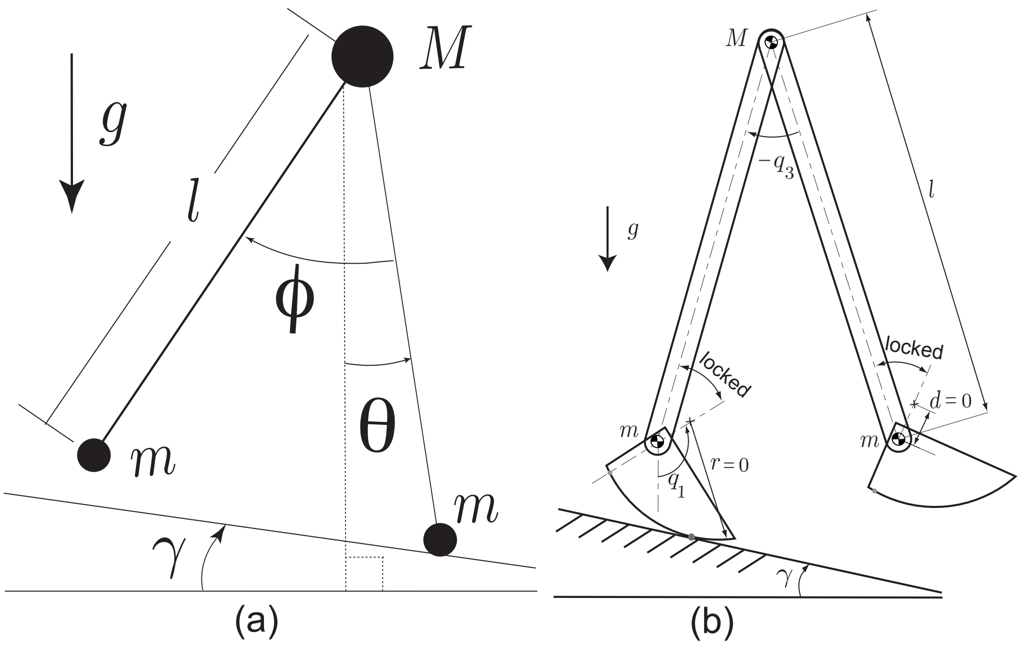


Figure 21: **2-D Simplest walker.** **(a)** Simplest walker analyzed by Garcia (figure source: Garcia's PhD thesis [5]), **(b)** Ranger model simplified to the simplest walker

earized map using central difference. We computed the biggest eigenvalues as $\sigma = 0.444444444411274$. Our Ranger-based eigen-value is accurate to 10th decimal place.

### 6.3 Reduction to the 2-D simplest walker

A more stringent comparison is with 'The Simplest Walker' which has a non-locked swing leg. Figure 21 (a) shows the 2-D simplest walker analyzed by Garcia [4]. The simplest walker has a point-mass $M$ at the hip. The legs of length $\ell$ are nearly massless, but there is a point-mass $m \ll M$ at the end of the swing leg. Garcia considered the limiting case, $m/M \to 0$. He found that the non-dimensional equations of motion are functions of a single parameter; the slope of the ramp $\gamma$. Garcia [4] does not report benchmark results for the simplest walker. So, using the equations derived by him and for a slope of $\gamma = 0.009$, we computed the fixed points and eigenvalues. These values are reported here and are used as benchmarks.

**Ranger model reduced to the simplest walker.** To derive the equations of motion we proceed as follows. First, we assume the gait sequence; single stance phase followed by heel-strike phase followed by single stance phase and so on. Next, we draw the free body diagram for Ranger model shown in figure 21 b). Finally, we use angular momentum balance about appropriate points to derive equations of motion.

Alternately, the equations of motion can be obtained from Ranger's equations of motion presented earlier. The equation of single stance are obtained from 11 with $i = P_1, H$. The equation for heel-strike is obtained from equations 21,24 and 26 and 24.

$$\begin{array}{lcc}
\text{Single stance (continuous):} & \dot{\overrightarrow{H}}_{/P_1} = \overrightarrow{M}_{/P_1} & \dot{\overrightarrow{H}}_{/H} = \overrightarrow{M}_{/H} \\
\text{Heel-strike (instantaneous):} & q_1 = r_1 - r_3 & q_3 = -r_3 \\
 & \overrightarrow{H}^{+}_{/P_1} = \overrightarrow{H}^{-}_{/P_2'} & \overrightarrow{H}^{+}_{/H} = \overrightarrow{H}^{-}_{/H'}
\end{array}$$

Table 1 gives the parameters for the reduced Ranger model to match up with Garcia's simplest walker model.

**Comparison of Ranger's reduction to the simplest walker.** We analyzed the simplified Ranger model using the parameters given in table 1 and using the recipe presented in section 6.1.

**Comparison of fixed points.** First using MATLAB's ODE45 (mixed 4th and 5th order Runge Kutta algorithm) with integration and root finder tolerance of $10^{-13}$, and with Garcia's equations of motion [5], we calculated a fixed point, $(\theta^*, \dot{\theta}^*, \phi^*, \dot{\phi}^*) = (0.200310900544287, -0.199832473004977, 0.400621801088574, -0.015822999948318)$.

Next, using the same tolerances for integration and root finder, but with Ranger's reduced equations of motion, we re-calculated the fixed points. We found the fixed point to be, $(q_1^*, \dot{q}_1^*, q_3^*, \dot{q}_3^*) = (0.200310750572992, -0.199832546623645, 0.400621501145995, -0.015822982402157)$. The fixed points differ in the 6th decimal place. This is consistent with the $m/M = 10^{-6} \neq 0$ that we used.

**Eigenvalues.** Using the fixed point from Garcia's equations of motion and using a perturbation of $10^{-5}$, we calculated the Jacobian of the linearized map using central difference. The non-zero eigenvalues of the linearized map were found to be, $\sigma_1 = -0.190099639087901 + 0.557599274928213i$ and $\sigma_2 = -0.190099639087901 - 0.557599274928213i$.

Using Ranger's reduced equations of motion and the above method to calculate the Jacobian, we got the following non-zero eigenvalues, $\sigma_1 = -0.190106213101483 + 0.557586570259502i$ and $\sigma_2 = -0.190106213101483 - 0.557586570259502i$. The eigenvalues agree to 3 decimals.

# 7 System identification for mechanical parameters

For numerical simulation we need to estimate the ten parameters shown in figure 1. CAD drawings could have been used to estimate most of these. But because there were many modifications not in the original drawings (rubber feet, glue, tape, etc) we took inertial parameter identification as a separate measurement project. The length parameters (ankle eccentricity $d$, the radius of feet $r$ and leg length $\ell$) are measured with a tape measure. Measurement of the inertial parameters is presented in section 7.2. Measurement of the spring parameters, the hip spring constant $k_h$ and the ankle spring constant $k_s$, is presented below (section 7.3). However, first we show how we dynamically balance the the legs, thereby making the robot symmetrical and simplifying the controller design and simulations.

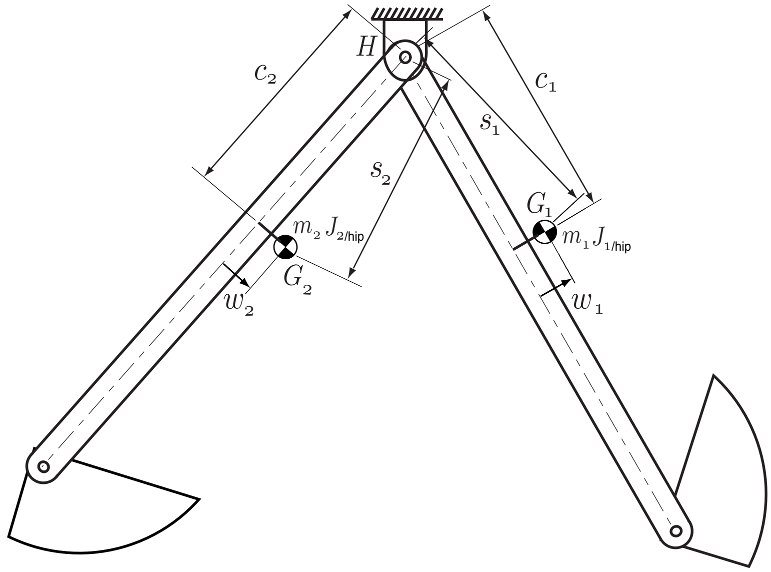


Figure 22: **Dynamic balance of the legs.** The two legs of the robot are dynamically symmetrical ('balanced') if they they have the same distances between the hip and ankle hinges, the same feet shapes and have 3 matching inertial properties. The two masses do not have to be the same, however.

## 7.1 Dynamic balancing of legs

We would like the legs to be dynamically balanced so that a controller that treats the legs as equal does not lead to a limping gait. Assuming a balanced machine, we can then simplify both the simulations and the controller by only dealing with a single step (rather than a 2-step sequence) as the basic action.

Figure 22 shows that the robot that may have unbalanced legs; i.e., the masses ($m_i$), inertia about hip ($J_{i/hip}$) and the location of the COM ($s_i$) of the two legs may different. Here $i = $ *inner* or *outer*.

**Number of parameters.** In side view the robot has 4 serial links (inner foot, inner leg, outer leg, outer foot). As detached planar rigid objects each of these has 4 inertial parameters: center of mass position relative to landmarks (e.g., hinges and corners) on the object ($x$ and $y$ positions using an object-based coordinate system); mass $m$; and moment of inertia about the center of mass $J_G$. Thus one could imagine up to 16 inertial parameters in the model. Neglecting the mass of the feet eliminates 8 of these (4 for each foot), reducing the number of inertial parameters to 8.

**Redundancy of parameters.** At every joint we can imagine adding and subtracting point masses. For example we could add a point mass $m_{add}$ to the inner legs at the hip and simultaneously add a negative mass $-m_{add}$ to the outer legs (that such is non-physical does not detract from the argument). No term in any of our equations of motion are affected by this addition and subtraction. Thus no motion nor torque is altered. One could also make the claim by appeal to Lagrange's equations: the expression for the system kinetic and potential energies are unaltered by this addition/subtraction.

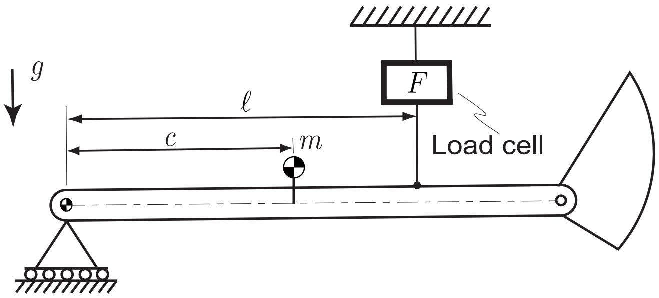


Figure 23: **Experiment to measure the first mass moment $c \cdot m$.** The robot is hinged at the hip joint and held such that the leg axis is perpendicular to gravity. Balance of moments about the hip hinge gives $mc = F\ell$.

We have thus changed the masses of each of the links and the locations of the centers of mass of each of the legs, but we have not changed any of the dynamics of the linkage. [As an aside, with this mass addition/subtraction we have changed the reaction forces transmitted at the hinge, but these do not affect the motions or the joint torques.] Thus, the supposed 8-dimensional parameter space is indifferent to one dimension. That is, there must be a collection of 7 parameters which can predict all coefficients in the governing equations. We claim that the following is such a set of 7 ($c$ and $w$ are local object-referenced $x$ and $y$ coordinates):

$$J_{1/hip},\ J_{2/hip};\qquad m_1c_1,\ m_2c_2;\qquad m_1w_1,\ m_2w_2;\qquad \text{and}\qquad M_{tot} \equiv m_1 + m_2. \tag{28}$$

Each of these parameters is indifferent to (doesn't change with) the hip-mass addition and subtraction described above. They are also independent in that by appending the list with another single number, for example the mass of one leg, all 8 of the original inertial parameters can be found ($m_i, J_{Gi}, c_i$ and $w_i$). In summary, the 7 mass parameters are: The first mass-moment of the mass distribution of each leg about the hip (2 numbers for each leg), the second polar mass moment of each leg about the hip (polar moment of inertia about the hip), and the total mass of the robot.

## 7.2 Measuring inertial parameters

Not coincidentally, the independent set of 7 parameters are exactly what it is possible to measure without disassembling the robot. Note, for example, that it is not possible to find the mass of one leg ($m_1$ or $m_2$). Nor is it possible to find the distance of the center of mass of one leg from the hip (can't find $s_1$ or $s_2$). We find the 7 parameters as follows:

1) The total mass $M_{tot}$ is found by weighing the robot.

2,3) The first moment of mass along each leg $c \cdot m$ is found for each leg by the experiment shown in figure 23.

4,5) The angle of the radial line from the hip on which the center of mass lies is found by hanging each leg from the hip hinge and measuring the angle $\theta$ of the leg (figure 24. Because $\tan\theta = w/c$ we have for each leg that $wm = \tan\theta\, cm$.

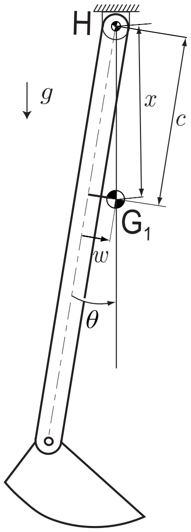


Figure 24: **Experiment to measure the fore-aft offset of the COM.** The robot is hinged at its hip joint $H$ and held in the vertical plane. In equilibrium the center of mass of the leg $G_1$ is directly below the hinge point $H$. The angle $\theta$ can be measured.

6,7) The inertia of each leg about the hip joint can be found from timing the small oscillations of each leg freely swinging from a hip clamped in place.

$$T_{\text{pend}} = 2\pi\sqrt{\frac{J_{/hip}}{gsm}} \tag{29}$$

where $sm = \sqrt{(cm)^2 + (wm)^2}$. So, for each leg,

$$J_{/hip} = \frac{T_{pend}^2}{4\pi^2} g\sqrt{(cm)^2 + (wm)^2}. \tag{30}$$

Thus all 7 independent inertial properties were measured without robot disassembly.

**Balance of 3 inertial parameters.** With these 7 independent parameters, symmetry of the two legs is achieved by making these 3 matches:

$$\begin{aligned} J_{1/hip} &= J_{2/hip} \\ m_1c_1 &= m_2c_2 \\ m_1w_1 &= m_2w_2 \end{aligned} \tag{31}$$

As mentioned, although we have four parameters for each leg ($J_{i/G}$, $m_i$, $c_i$ and $w_i$) there are only three conditions that ensure dynamic balance of the two legs.

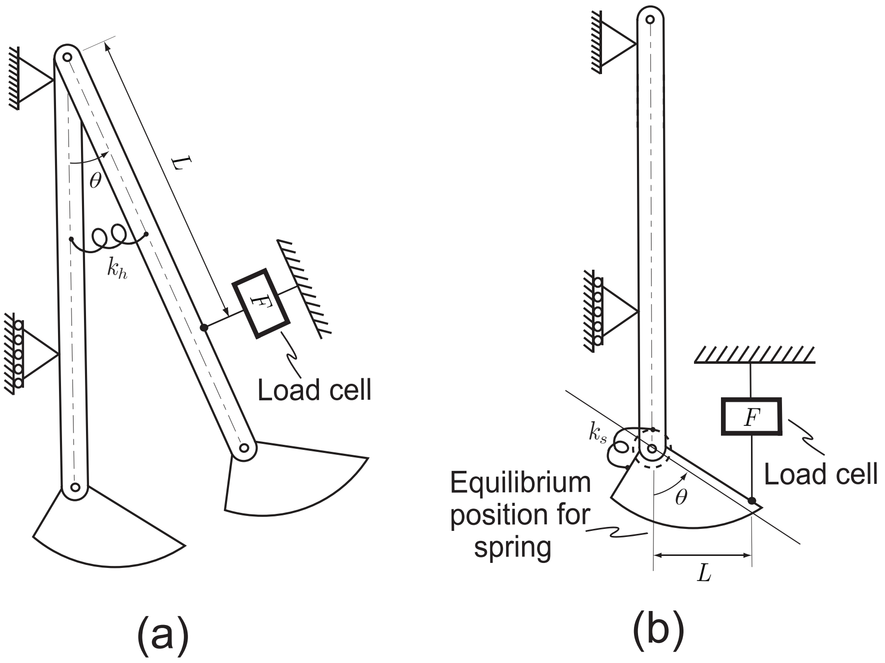


Figure 25: **Measuring the spring constants. a)** The hip spring stiffness (see figure 6 on page 12 and related text) is determined by measuring the force to deflect the leg a given angle. **b)** The ankle spring constants (the elasticities of the cable drive, see figure 3 on page 9 and related text) are measured by locking the ankle motors and measuring the force to deflect the ankles.

**Physical balancing.** We made the best balance we could by appropriate placement of the batteries on the outer legs. We chose not to add additional masses. Because we were constrained in our battery attachment points we could not get perfect dynamic balance (i.e., could not achieve equality within measurement accuracy of equation 31).

### 7.3 Measuring spring constants

**Measuring the hip spring constant.** Figure 25a shows the set up used to estimate the hip spring constant. In this experiment, the robot was placed in the horizontal plane and so gravity does not influence the experiment. The hip springs are designed so they have no torque when the legs are parallel to each other (see figure 6 on page 12). First, we checked this by noting that the spring torque is zero when the legs are parallel. Next, we pulled Ranger's legs with a digital 'fish' scale (a load cell with hooks) and noted the hip angle $\theta$. The hip torsional spring constant is then

$$k_h = FL/\theta.$$

As predicted (see figure 6) the spring is nearly linear in Ranger's operating region (hip angle ±0.5 rad).

**Measuring the ankle spring constant.** Figure 25b shows the ankle spring-constant measurement. The motor was put in position control (locked at a fixed angle) and the foot deflected with

| Parameter | Value |
|---|---|
| $\ell$ | 0.96 m |
| $r$ | 0.2 m |
| $d$ | 0.11 m |
| $wm$ | 0.0 |
| $cm$ | 0.72 kg m |
| $J_{hip}$ | 0.55 kg m$^2$ |
| $k_h$ | 7.6 N m/rad |
| $k_s$ | 14 N m/rad |

Table 2: Values of robot parameters. These were measured/estimated in bench tests.

the fish scale (see Figure 25b) and the angle of ankle deflection $\theta$ measured. Thus the ankle spring constant

$$k_s = FL/\theta.$$

Although this measurement includes a parallel contribution from the soft foot return spring (see figure 3), it is dominated by the stiffness of the ankle drive cable (the Achilles tendon).

### 7.4 Summary of parameters estimated

We summarize the various parameters estimated in this section.

## 8 System identification for motors and gearboxes

In section 3.3 in the main paper, we showed the motor and gearbox model we used for simulation and optimization. First here, we re-state these equations and follow that with a discussion on how the model was found and the parameters identified.

### 8.1 Power equation

The power equation gives the power consumption ($P$) as a function of the current ($I$) and the speed ($\omega$). In the end we wrote this equation as

$$P = VI = \{IR + V_c \operatorname{sgn}(I) + GK\omega\} I \tag{32}$$

Here the gear ratio $G$ is known based on manufacturer's specifications and we have to identify the three constants: resistance $R$, contact voltage $V_c$ and torque constant $K$.

### 8.2 Torque equation

The torque equation gives the output shaft torque ($T$) as a function of the current and shaft speed. The function $T_f(I, \omega)$ in the expression below is the frictional torque and we have assumed that it

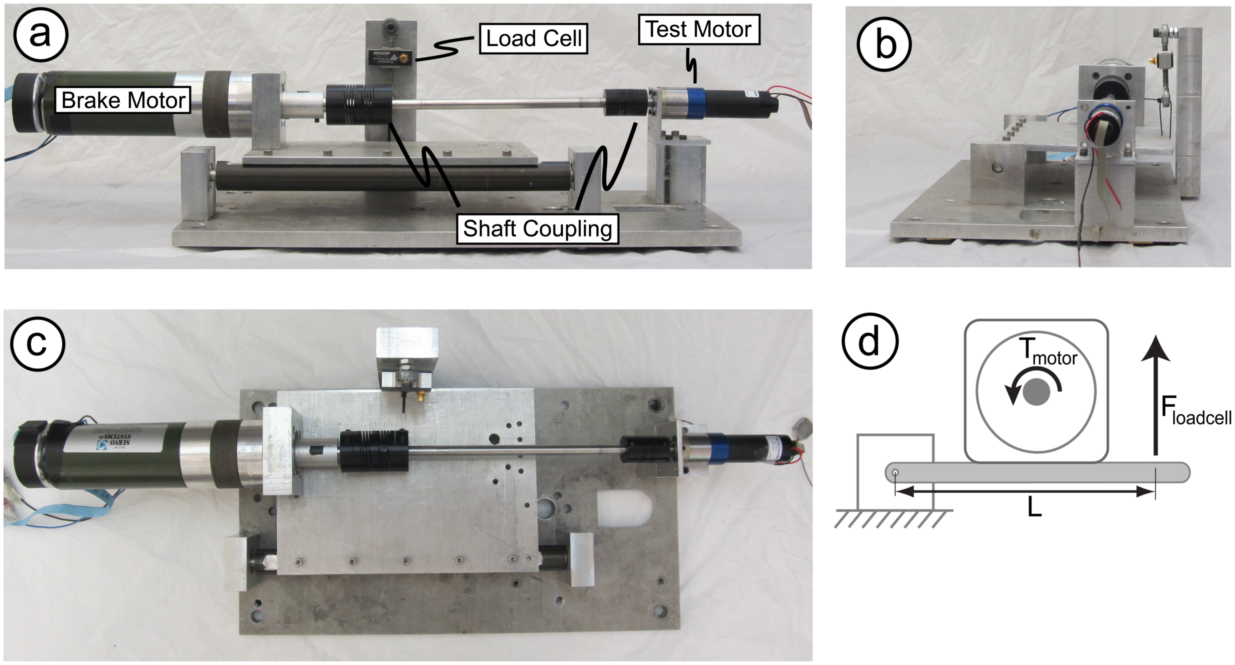


Figure 26: **Cantilever set up used for system identification for DC motors.** A 'brake' motor is mounted on a hinged plate so the torque acting on it can be measured. It is driven by the motor being tested, which is mounted on a solid workbench, via two helical shaft couplings and a slender steel rod. A DC power source is connected to the test motor that maintains the input voltage $V$. By varying the current in the brake motor, varying braking torques can be applied to the test motor. Test motor current $I$, output shaft speed $\omega$, and output shaft torque $T$ are measured and used for system identification. Note that the 'brake' motor can also be powered so that the test motor can be characterized in the negative-work regime (i.e., as a generator).

depends on the current ($I$) and shaft speed ($\omega$).

$$T = G(KI - J_m \overbrace{G\dot{\omega}}^{\text{motor acceleration}}) - T_f(I,\omega) \tag{33}$$

We found that the friction torque $T_f(I,\omega)$ can be reasonably decomposed into a constant friction term and viscous friction term.

$$\begin{aligned} |T_f(I,\omega)| &\le C_{0s}(I) && \text{if } \omega = 0 \\ T_f(I,\omega) &= C_1\omega + C_{0d}(I)\ \mathrm{sgn}(\omega) && \text{otherwise} \end{aligned} \tag{34}$$

where $C_1$, $C_{0s}$ and $C_{0d}$ are coefficients of viscous, static and dynamic friction respectively and the latter two are assumed to be current dependent.

Next, we found reasonable fit using $C_{0s}(I) = C_{0s} + C'_{0s}|I|$ and $C_{0d}(I) = C_{0d} + C'_{0d}|I|$. We characterize the current dependent part of the constant friction by parameter $\mu$ as follows. We put

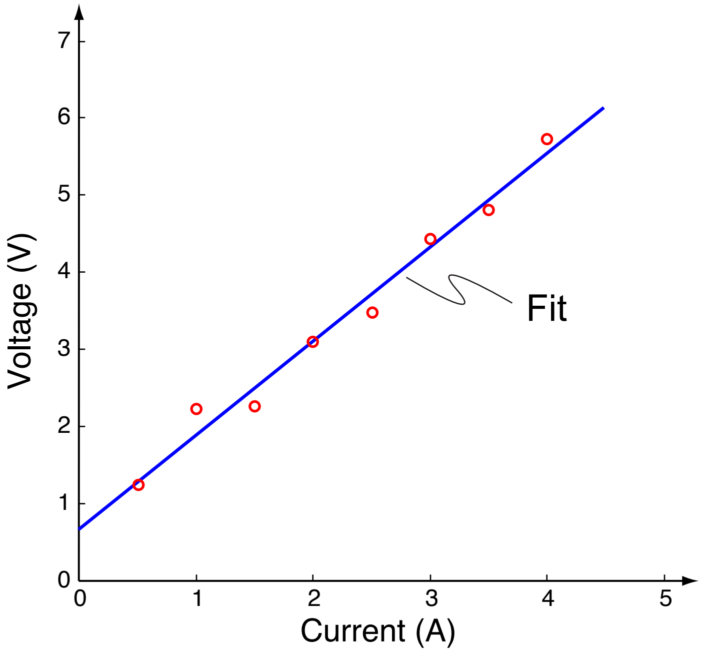


Figure 27: **Curve-fitting for contact voltage and terminal resistance.** Least squares curve-fitting voltage and current data for a stalled motor gives a terminal resistance $R = 1.3\ \Omega$ (slope) and contact voltage drop $V_c = 0.7\ V$ (y-intercept).

$C'_{0s} = \mu_s GK|I|$ and $C'_{0d} = \mu_d GK|I|$. Thus our frictional torque becomes,

$$|T_f(I,\omega)| \leq C_{0s} + \mu_s GK|I| \qquad \text{if } \omega = 0 \qquad (35)$$

$$T_f(I,\omega) = C_1\omega + C_{0d}\ \mathrm{sgn}(\omega) + \mu_d GK|I|\ \mathrm{sgn}(\omega) \qquad \text{otherwise} \qquad (36)$$

In the above equations, the gear ratio $G$ is known based on manufacturer's specifications. The torque constant $K$ is known from the system identification on the power equation presented earlier. Thus in equation 35 and 36, we have to identify the five constants, $C_{0s}$, $C_{0d}$, $\mu_s$, $\mu_d$ and $C_1$.

### 8.3 Cantilever test set-up for data collection

**Details of set-up.** Figure 26 shows the labeled photograph of our experimental set-up. The set-up consists of two motors; a Faulhaber test motor of the type used on the robot and a Maxon brake motor used as the motor load. Each motor can be individually controlled by DC power supplies. The test motor is integrated with a 14:1 gear box. The motors are connected to each other by two helical shaft couplings and a slender steel rod. A Hall-effect current sensor measures the current flowing through the test motor. A rotary encoder measures the angular position of the brake motor. Angular position can be converted to angular speed by finite differencing and low-pass filtering. Using the brake motor gear ratio and its measured angular speed, we can find the speed of the output shaft, the slender steel rod in the set-up. A load cell (Transducer Techniques, MLP-75) measures the test motor torque output. By varying the current flowing through the braking motor,

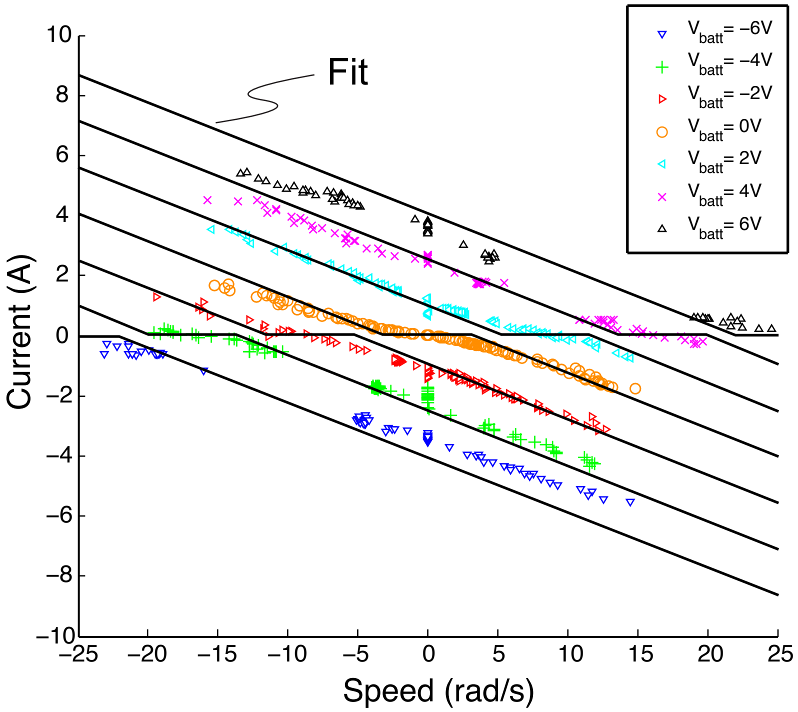


Figure 28: **Curve-fitting for motor torque constant.** Motor torque constant $K = 0.018$ V s/rad was curve-fitted from equation 32 using motor current $I$, output shaft speed $\omega$ and DC voltage $V$ data obtained from the cantilever set-up. In equation 32 we used gear ratio $G = 14$ as per manufacturer's specification. Constants $R = 1.3\ \Omega$ and $V_c = 0.7\ V$ were obtained in an earlier experiment (see figure 27).

varying braking torques can be applied. Data is recorded by a National Instruments LabVIEW program which is interfaced with the sensors through a data acquisition system.

**Data collection.** The data collection was done as follows.

- A DC voltage ($V$) was set on the test motor. The braking torque was varied by varying the current to the brake motor. Test motor current ($I$), output shaft speed ($\omega$) and output shaft torque ($T$) reading were noted for increasing and decreasing braking torques.
- A different motor voltage was fixed and the test repeated. The test motor voltages chosen were $-6, -4, -2, 0, 2, 4$ and $6\ V$.

## 8.4 Fit the power equation

In order to fit the power equation 32 we need to find the constants $R$, $V_c$ and $K$. First, we stall the motor and noting the current at various voltages, we fit the constants $V_c$ and $R$. Next, using the data from the cantilever experiment we fit the constant $K$.

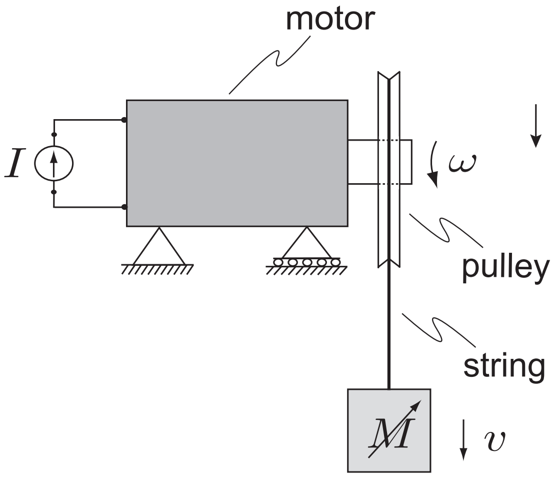


Figure 29: **Measuring friction coefficients.** In some experiments the motor was initially still, in others it is turning at known rate.

### 8.4.1 Fit the resistance ($R$) and contact voltage drop ($V_c$) by stalling the motor

In equation 32, we first set to identify the resistance $R$ and contact voltage drop $V_c$. We stalled the motor and applied various currents $I$ and measured the output voltage $V$. Putting speed $\omega = 0$ in equation 32 we have $V = IR + V_c\ \mathrm{sgn}(I)$. Using a least squares fit, we found $R = 1.3\ \Omega$ and $V_c = 0.7\ V$. Results of the fit are shown in figure 27.

### 8.4.2 Fit the the torque constant ($K$) from cantilever experiment

Having fitted the resistance $R$ and contact voltage drop $V_c$ in equation 32, we only need to fit the torque constant $K$. Using the DC voltage $V$, motor current $I$ and output shaft speed $\omega$ data from the cantilever experiment and knowing that the gear ratio $G = 14$, we fitted the torque constant $K = 0.018$ as shown in figure 28.

## 8.5 Fit the torque equation

In order to fit the torque equations 35 and 36, we have to identify the five constants, $C_{0s}$, $C_{0d}$, $\mu_s$, $\mu_d$ and $C_1$. First, using a series of pulley experiment as shown in figure 29 we identify all the five constants. Next, using the data from the cantilever experiment we check our fit and also point out the necessity of having the $|I|$ function in the frictional torque equations 35 and 36 .

### 8.5.1 Fit the dynamic friction at zero current ($C_{0d}$)

We set the motor in figure 29 in open loop ($I = 0$) and increased the mass $M$ in increments of 5 gram while gently tapping the pulley until it set into slight motion. At about $M = 40$ gram, the mass started to move downward (clockwise rotation of pulley when viewed from the right side). Using the radius of the pulley ($r = 2.5$ cm), mass $M$, gravity $g$ and equation 33 and equation 36, we calculated the dynamic friction at zero current to be $T = Mgr = T_f = C_{0d} = 0.01\ Nm$. An

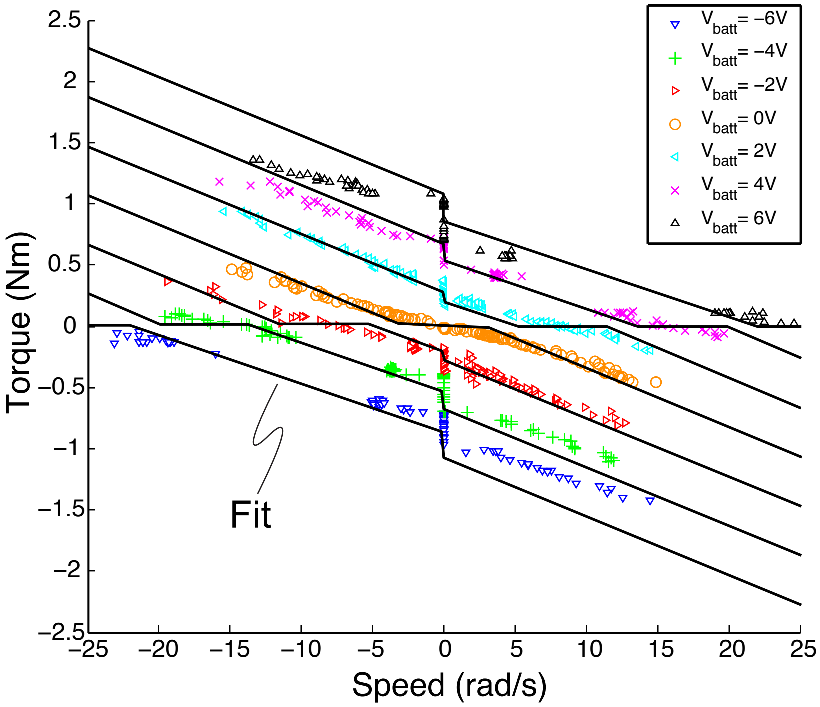


Figure 30: **Checking the friction model.** The frictional torque identified using a series of pulley experiments is checked with data obtained from the cantilever experiment. Various shaped dots are data and the curves are the fit. The positive work regimes are where the torque and angular velocity have the same signs (first and third quadrants). The worst data fits are for high braking torques (lower right and upper left on plots). The discontinuities at speed = 0 are from friction force reversals. The discontinuities near the torque = 0 axis are due to the reversing of the contact voltage drop when the current reverses.

identical value for the dynamic friction was calculated when the mass was hung on the other side of the pulley (counter-clockwise rotation of pulley when viewed from the right side).

### 8.5.2 Fit the static friction at zero current ($C_{0s}$)

We set the motor in figure 29 in open loop ($I = 0$) and increased the mass $M$ in increments of 5 gram till the motor-pulley, with no tapping, until it set into slight motion. At about $M = 45$ gram the mass began to move downwards. Using the radius of the pulley $r$, mass $M$, gravity $g$ and equations 33 and equation 35, we calculated the static friction at zero current to be $T = Mgr = T_f = C_{0s}(0) = 0.01$ N m. The same static friction value was calculated when the test was repeated with the mass hanging on the other side of the pulley.

### 8.5.3 Fit the viscous friction at zero current ($C_1$)

We set the motor in figure 29 in open loop ($I = 0$) and increased the mass to $M_1$ until the motor-pulley set in motion. We noted that speed $\omega_1$. Next we changed the mass to $M_2$ and repeated the experiment and noted the new speed $\omega_2$. From equation 33 and equation 36 we have

$$T_1 = M_1 gr = T_{f1} = C_1\omega_1 + C_{0d}$$
$$T_2 = M_2 gr = T_{f2} = C_1\omega_2 + C_{0d}$$

Subtracted the first equation from the second gives,

$$C_1 = \frac{(M_2 - M_1)gr}{\omega_2 - \omega_1} \tag{37}$$

Using equation 37, we calculated the viscous friction to be $3.3 \times 10^{-3}$ N m s/rad. The same experiment when repeated with the mass hung on the other side of the pulley gave a similar value.

### 8.5.4 Fit the coefficient of dynamic friction ($\mu_d$)

We repeated the experiment in section 8.5.1, except that we set the current to a non-zero value. If $M$ is the load at which the motor-pulley system just starts to move then, from equation 33 and equation 36, just at the onset of motion we have

$$T = Mgr = GKI - \mu_d GK|I| - C_{0d}$$

Solving for $\mu_d$ gives,

$$\mu_d = \frac{1}{\mathrm{sgn}(I)}\left\{1 - \frac{Mgr + C_{0d}}{GKI}\right\} \tag{38}$$

We repeated this test for different current values and also by putting the mass on the other side of the pulley. The average value for $\mu_d$ across various tests was found to be 0.1.

### 8.5.5 Fit the coefficient of static friction ($\mu_s$)

We repeated the experiment in section 8.5.2, except that we set the current to a non-zero value. Again, if $M$ is the load at which the motor-pulley system just moves when tapped then, from equation 33 and equation 35, just at the onset of motion we have

$$T = Mgr = GKI - \mu_s GK|I| - C_{0s}$$

Solving for $\mu_s$ gives,

$$\mu_s = \frac{1}{\mathrm{sgn}(I)}\left\{1 - \frac{Mgr + C_{0s}}{GKI}\right\} \tag{39}$$

We repeated this test for different current values and also by putting the mass on the other side of the pulley. The average value for $\mu_s$ across various tests was found to be 0.1. Finally, we used the value of the constants $C_{0s}$, $C_{0d}$, $C_1$, $\mu_s$ and $\mu_d$ and checked the torque equation with the data obtained from the cantilever experiment. Figure 30 shows our model fit with the torque-speed-voltage data obtained from the cantilever experiment.

| Parameters | Symbol | Expts. (Specs.) |
|---|---|---|
| Terminal resistance ($\Omega$) | $R$ | 1.3 (0.7) |
| Contact voltage drop (V) | $V_c$ | 0.7 |
| Torque constant (N m/A) | $K$ | 0.018 (0.017) |
| Viscous friction (N m s/rad) | $C_1$ | 0 |
| Constant friction (N m) | $C_0 = C_{0s} = C_{0d}$ | 0.01 |
| Current-dependent const. friction | $\mu = \mu_s = \mu_d$ | 0.1 |

Table 3: Comparison between experimental values with manufacturer's specification for the motor model. The static and dynamic constant friction terms have the same value, i.e. $C_{0s} = C_{0d}$ and hence these are replaced with the term $C_0$. Similarly, static and dynamic current dependent friction terms have the same value, i.e. $\mu_s = \mu_d$ and hence these are replaced with the term $\mu$. Note that the measured resistance is almost twice that reported in the specification sheet. Also, the brush-commutator contact voltage drop of the motor is not mentioned in the specification sheet

### 8.6 Summary of constants for motor model

Table 3 shows all the constants obtained in the motor equation.

All of the bench tests were with a 14:1 gearbox. On the robot we used a 43:1 reduction for the ankles and 66:1 for the hip. These motors and gearboxes should have at least slightly different friction parameter values. However, we did tests of robot leg swing and of foot flip up and down and found that the parameters above gave sufficiently accurate predictions.

## 9 Smoothings for simulations and optimizations

Because of the reversals of contact voltage at current reversals and of friction force at velocity reversals various terms in the simulations and optimizations are not smooth.

There is some subtlety in the reason for the lack of smoothness in the differential equations to survive to the optimization objective function. If the discontinuities were simply crossed they would not lead to discontinuities in the objective function. But because the optimal trajectories tend to sit on the discontinuities for extended times, not just passing through them, they do survive to the optimizations.

To eliminate the related numerical issues, especially problems with convergence of the optimization software, we smooth such discontinuities.

The smoothings are also needed in our benchmark optimizations based on mechanical power, which, assuming no regeneration, has a discontinuity at $x = 0$.

**Smoothing for $[x]^+$.** Steps (Heaviside functions) in voltage and force only lead to ramps (integral of Heaviside function) in the electrical power (which is integrated to obtain our objective function). So our main concern is with smoothing ramp functions. The unit ramp function $[x]^+$ is zero for negative $x$ and simply $x$ for positive $x$:

$$[x]^+ = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{40}$$

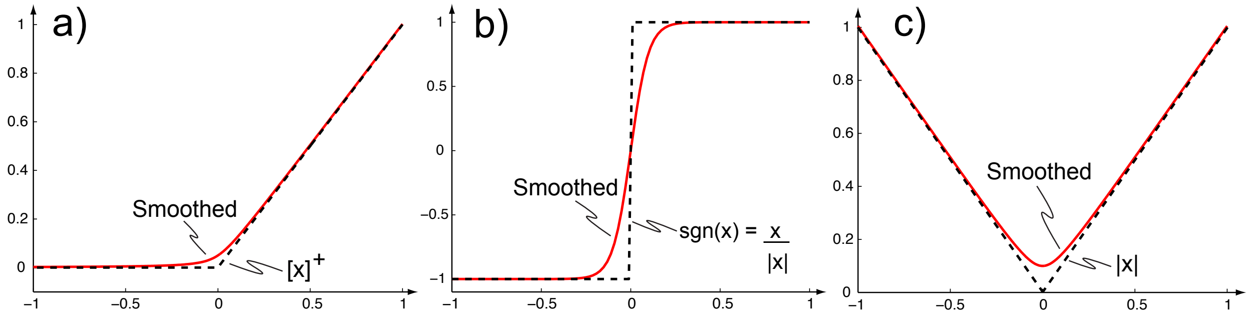


Figure 31: **Smoothings for discontinuous functions.** The smoothings for a unit ramp function, a step function and for the absolute value function (dotted black lines= function, solid red lines = our approximation).

The function has a kink at $x = 0$. A smooth approximation to $[x]^+$ is given in by [10, 12].

$$[x]^+ \approx \frac{x + \sqrt{x^2 + \epsilon_1^2}}{2} \tag{41}$$

As a rule we use $\epsilon_1 = 0.01$. Figure 31a) compares the smooth approximation with the actual function.

**Smoothing for sgn**$(x)$**.** When needed we also smooth the signum function $\mathrm{sgn}(x)$:

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{42}$$

The function is discontinuous at $x = 0$. A smooth, continuous approximation of $\mathrm{sgn}(x)$ is in [3].

$$\mathrm{sgn}(x) \approx \tanh\left(\frac{x}{\epsilon_2}\right) \tag{43}$$

We generally use $\epsilon_2 = 0.01$. Figure 31b) compares the smooth approximation with the actual function.

**Smoothing for** $|x|$**.** The absolute value function can be defined as:

$$|x| = \begin{cases} x & \text{if } x > 0 \\ -x & \text{if } x \leq 0 \end{cases} \tag{44}$$

This function also has a kink at $x = 0$. A smooth approximation to $|x|$ is given in equation by [10, 12] as

$$|x| \approx \sqrt{x^2 + \epsilon_3^2} \tag{45}$$

Again we generally us $\epsilon_3 = 0.01$. Figure 31c) compares the smooth approximation with the actual function.

# 10 Benchmarks for optimal trajectory control

The trajectories used on Ranger are based on optimizations. To check the veracity of our optimization calculations we optimize situations in which we already have some understanding of the optima and which also can be made to conform to the overall Ranger governing equations and framework, namely:

1. passive dynamic walking [7], and
2. point-mass energy-optimal walking [11].

These tests are explained below.

## 10.1 Discovering passive dynamic walking

McGeer showed that a 2D 2-legged robot with suitable mass distribution can walk down a shallow slope with no actuation [7]. For the first validation of the Ranger simulation we keep (and optimize) the hip motor (but neglect friction). To keep this check simple we locked the ankles. We know from previous passive-dynamics research that on a small slope this model has periodic solutions with zero hip torque and thus, with a simplified motor model, zero hip current. We also know that for the motor model the minimum conceivable energy use is zero, with zero current and thus zero torque at all times. Thus the optimal control solution should be one with zero hip current for all

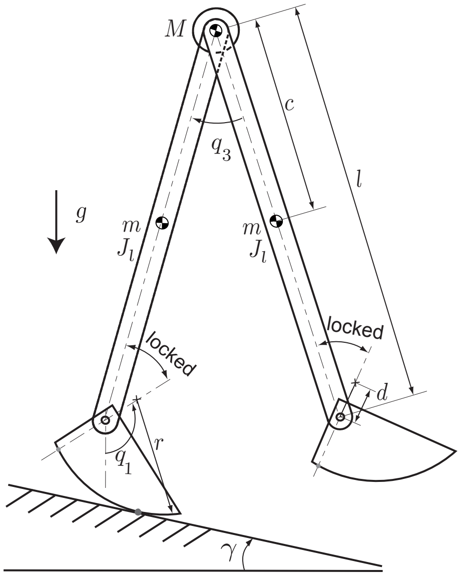


Figure 32: **Passive dynamic walking validation.** Ranger with ankles locked on a shallow slope. The hip motor can power walking.

| Parameter | Value |
|---|---|
| $\ell$ | 0.96 m |
| $r$ | 0.2 m |
| $d$ | 0.11 m |
| $k_h$ | 0 |
| $k_s$ | Not in equation |
| $J_\ell$ | 0.24 $\mathrm{kg\,m^2}$ |
| $m$ | 2 kg |
| $M$ | 4.5 kg |
| $w$ | 0 m |
| $c$ | 0.3 m |
| $g$ | 10 $\mathrm{m/s^2}$ |
| $\gamma$ | 0.005 |
| $C_{1FW}$ | 0 |
| $C_{2FW}$ | Not in equation |

Table 4: Ranger parameter values for discovering passive dynamic walking.

time, namely passive-dynamic walking. Here is how we checked for the optimization's ability to make this discovery.

**Model used for 'discovery' of passive dynamic walking.** The robot model is shown in figure 32. The equations of motion are simplified from Ranger's governing equations. The equation of single stance is obtained from 11 by putting $i = P_1, H$. The equation for heel-strike is obtained from equations 21, 24 and 26. Note that we *a priori* preclude a double-stance phase.

For this optimization we use a simplified (i.e., frictionless) motor model

$$\begin{aligned} &\text{Power model:} && P = I^2R + G_H K I \omega \\ &\text{Torque model:} && T = G_H K I \end{aligned}$$

where the hip power is $P$, the hip torque is $T$, hip motor current is $I$, hip motor speed $\omega$, motor resistance is $R$, motor constant is $K$, and hip motor gear ratio is $G_H$. Note that this model is torque-free when the electrical power (and current) is zero.

**Optimization problem for discovery of passive dynamic walking.** We seek a control strategy (hip motor current as a function of time) and initial conditions, that minimize the cost

$$\text{COT} \;=\; \frac{\text{Energy used per step}}{\text{Weight} \times \text{Step length}} \;=\; \int_{t=0}^{t=t_{\text{step}}} \frac{|P|dt}{M_{\text{tot}}\; g\; d_{\text{step}}} \tag{46}$$

subject to the following constraints: periodicity, namely that a the state vector at the beginning of single stance should be equal to the state vector just after heel-strike; and the vertical reaction forces on the grounded legs should be positive at all times.

The absolute value of hip power $P$ is not smooth, so we smooth it in such a way that our gradient-based optimization software will perform better (see section 9).

| Parameter | Value |
|---|---|
| $G_A$ | Not in equation |
| $G_H$ | 66 |
| $K$ | 0.017 N m/A |
| $R$ | 1.3 Ω |
| $V_c$ | 0 |
| $J_m$ | Not in equation |
| $\mu_H$ | 0 |
| $\mu_A$ | 0 |
| $C_{H1}$ | 0 |
| $C_{H0}$ | 0 |
| $C_{A1}$ | 0 |
| $C_{A0}$ | 0 |
| $P_{\text{fixed}}$ | Not in equation |

Table 5: Motor parameter values for discovering passive dynamic walking. Symbols A and H denote ankle and hip respectively.

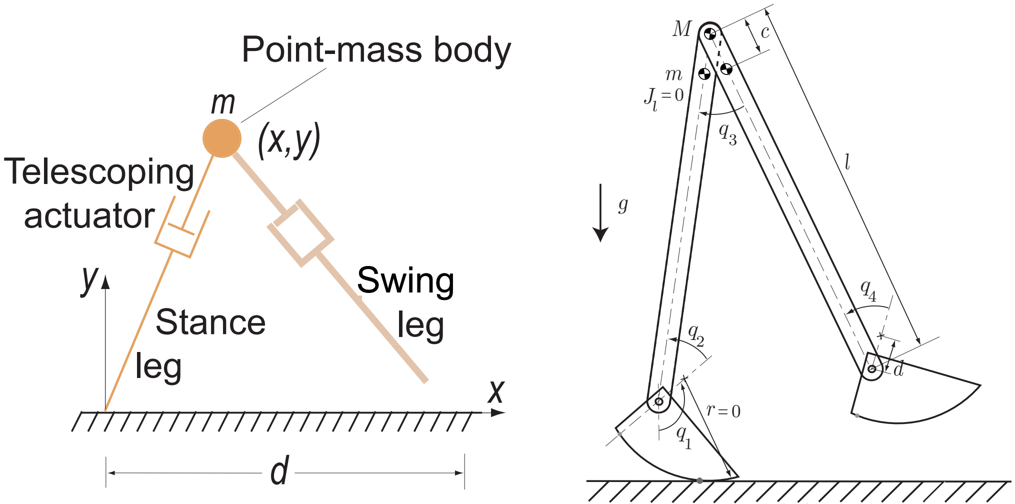
**Parameterization of the optimal trajectory.** The optimization parameters are:

- The state at beginning of single stance ($\chi^i_{ss}(t=0)$); $\chi^i_{ss}(t=0) \;=\; [q_1\ ,\ q_3,\ \dot{q}_1\ ,\ \dot{q}_3]_{ss}(t=0)$.
- Step time ($t_{\text{step}}$)
- The currents in the hip motor in single stance ($I(t)$). We assume piecewise linear-in-time currents. In single stance, we divide time into $N$ intervals; $t_0, t_1, \ldots, t_N$. Here $t_{i+1}-t_i = 1/N$, $i = 0, 1, ..., N$, $t_0 = 0$ and $t_N = t_{\text{step}}$. This means we have $2(N+1)$ unknowns for currents in single stance; $I(t=t_0), I(t=t_1), \ldots, I(t=t_N)$.

**Constraints.** The optimization proceeds subject to various constraints on the optimization parameters and things calculated from those parameters:

- The periodicity constraints of the state at the beginning of single stance should match the state just after heel-strike; $\chi^i_{ss}(t=0) = \chi^+_{hs}(t=t_{\text{step}})$.
- The transition from single stance to heel-strike takes place when the swinging leg's foot hits the ground at time $t = t_{\text{step}}$, as expressed in equation 5 on page 41.
- Vertical ground reaction force for the foot on the ground in single stance should be positive and this is enforced at the grid points. This gives $N+1$ inequality constraints.

**Method of optimization.** With the initial conditions in single stance $\chi^i_{ss}(t=0)$, we integrate equations of motion for single stance till we reach the end of single stance at time $t = t_{\text{step}}$. The torques for use in the single stance equations are obtained from the motor's torque equation using the assumed piecewise linear currents in single stance. Integration is from grid point to grid point so there are no discontinuities during an integration step.



(a) (b)

Figure 33: **Point-mass level walking validation.** **a)** Point-mass model analyzed by Srinivasan (figure source: Srinivasan's PhD thesis [10]) **b)** Ranger model simplified to a point-mass model by various special parameter values. In particular the foot radius $r$ is set to zero, making the foot a point at a distance $d$ from the ankle.

After the integration reaches the end of single stance, we apply the heel-strike condition to get the state after heel-strike $\chi^+_{hs}(t = t_{\text{step}})$. For each integration of a full step the total energy cost is evaluated. The optimization software varies the various optimization parameters attempting to minimize this cost (equation 46), while respecting the various constraints described above. We used the sequential quadratic programming package SNOPT [6] to solve the optimization problem.

**Passive walking.** First, without any optimization, we looked for passive solutions for the set of Ranger's parameters assumed here using the root-finding procedure outlined in section 6.1. We found two such passive solutions. The first solution had a step time of $t_{\text{step}} = 0.79329202031544$, and fixed points of $(q_1^*, \dot{q}_1^*, q_3^*, \dot{q}_3^*) = (2.99716718892289, -0.54617378605291, -0.28885092932654, -0.53131730110868)$. The second solution had a step time of $t_{\text{step}} = 0.93624752862330$, and fixed points of $(q_1^*, \dot{q}_1^*, q_3^*, \dot{q}_3^*) = (2.96417643512919, -0.65228639005082, -0.35483243689880, -0.02749381544188)$. Generally people find two periodic solutions for the passive dynamic walkers [5, 7] for a given slope.

**Passive walking is discovered.** Next, we ran the trajectory optimization starting with initial guesses far from the passive solutions. Each time the optimization converged to one of the above two solutions and with zero current for all time. Thus the trajectory optimization successfully discovered passive-dynamic walking.

## 10.2 Discovering optimal level-ground walking of a point-mass model

Srinivasan, Ruina [11] and Srinivasan [10] present a point-mass legged locomotion model. Using energy-optimal trajectory control, they show that the model chooses to walk at low speeds, run at

| Parameter | Value |
|---|---|
| $\ell$ | 1 m |
| $r$ | 0 |
| $d$ | 0.05 m |
| $k_h$ | 0 |
| $k_s$ | Not in equation |
| $J_\ell$ | Not in equation |
| $m$ | 0.01 kg |
| $M$ | 1 kg |
| $w$ | 0 |
| $c$ | 0.5 m |
| $g$ | 1 m/s$^2$ |
| $\gamma$ | 0 |
| $C_{1FW}$ | 0 |
| $C_{2FW}$ | Not in equation |

Table 6: Ranger parameter values used to discover energy-optimal level walking.

fast speeds, and at intermediate speeds discovers a new kind of walk, which they call the pendular-run. For this benchmark we are only interested in the walking solution. In particular, the optimal strategy for walking is an impulsive push-off just before heel-strike followed by a stance phase consisting of motion as a simple inverted pendulum. For this second validation, we approximate the point-mass model by putting most of Ranger's mass on the hip, and making the legs light. We leave the foot eccentricity non-zero but make the foot zero radius. In effect this makes the leg extensible, running between the hip and the infinitesimal foot (which is not inline with the leg). We then see if the optimization discovers optimum level walking with an impulsive push-off just before heel-strike.

**Model for comparison with point-mass locomotion.** The model is shown in figure 33b. We obtain the equations of motion from Ranger's equations of motion. The equations for single stance are obtained from 11 with $i = P_1, A_1, H$. The equations for heel-strike are obtained from equations 21 to 26.

$$\begin{array}{lrl}
\text{Single stance (continuous):} & \dot{\overrightarrow{H}}_{/i} = \overrightarrow{M}_{/i} & \text{where } i = P_1, A_1, H \\
\text{Heel-strike (instantaneous):} & q_1 = r_1 - r_3 & q_3 = -r_3 \\
 & q_2 = r_4 & q_4 = r_2 \\
 & \overrightarrow{H}^{+}_{/P_1} = \overrightarrow{H}^{-}_{/P_2'} & \\
 & \overrightarrow{H}^{+}_{/A_1} = \overrightarrow{H}^{-}_{/A_2'} & \\
 & \overrightarrow{H}^{+}_{/H} = \overrightarrow{H}^{-}_{/H'} &
\end{array}$$

**Optimal trajectory control problem for comparison with the point-mass model.** We assume a walking sequence given by these phases: single stance, heel-strike, single stance, repeat. We seek a control strategy that includes finding initial conditions in single stance, and torque in the hip and stance ankle motor as a function of time, that minimize the cost given by,

$$\text{COT} = \frac{\text{Energy used per step}}{\text{Weight} \times \text{Step length}} \tag{47}$$

We use the following constraints: periodicity requires that the state vector at the beginning of single stance should be equal to the state vector just after heel-strike; step length and average step velocity are specified; and the vertical reaction forces on the grounded legs should be positive at all times. Because the solution we are trying to discover has infinite forces, the numerics are helped by constraining hip and ankle motor torques to be within given bounds. Convergence to the singular solution is inferred by the torques always using the bounds, no matter how high (as explained in [11]).

### 10.2.1 Numerical formulation of point-mass optimal trajectory problem

**Cost.** The cost is the energy metric cost of transport and is defined as,

$$\text{COT} = \int_{t=0}^{t=t_{\text{step}}} \frac{\{|T_2 u_2| + |T_3 u_3|\}dt}{M_{\text{tot}}\, g\, d_{\text{step}}} \tag{48}$$

The ankle and hip torques are $T_2$ and $T_3$ respectively. The ankle and hip speeds are $u_2$ and $u_3$ respectively. The absolute value is not a smooth function; it has a kink at 0. So we smooth this function as shown in section 9. The numerator in equation 48 is the mechanical power at the joints, counting negative and positive power equally.

**Parameters.** The optimization parameters are:

- State at beginning of single stance ($\chi^i_{ss}(t=0)$);
  $\chi^i_{ss}(t=0) \;=\; [q_1\;,\; q_2\;, q_3,\; \dot{q}_1\;,\; \dot{q}_2\;,\; \dot{q}_3]_{ss}(t=0)$.
- Note that we ignore foot scuffing altogether and that we do not model the motion of the foot attached to the swinging leg's foot. However, we do need to specify the touchdown angle of the swinging leg's foot ($q_4^{hs}$) at heel-strike and this is an optimization parameter.
- Step time ($t_{\text{step}}$)
- Torque in the motors in single stance ($T_2(t)$, $T_3(t)$). We assume piecewise linear torques. In single stance, we divide time into $N$ intervals; $t_0, t_1, \ldots, t_N$. Here $t_{i+1} - t_i = 1/N$, $i = 0, 1, ..., N$, $t_0 = 0$ and $t_N = t_{\text{step}}$. This means we have $2(N+1)$ unknowns for torques in single stance; $T_j(t=t_0), T_j(t=t_1), \ldots, T_j(t=t_N)$, where $j = 2, 3$.

**Constraints.** The optimization constraints are:

- Periodicity constraints: the state at the beginning of single stance should match the state just after heel-strike; $\chi^i_{ss}(t=0) = \chi^+_{hs}(t=t_{\text{step}})$.

- Step velocity (average over a step) is specified, $v_{\text{step}} = V$.
- Step length is specified, $d_{\text{step}} = D$.
- Transition from single stance to heel-strike takes place when the swinging leg's foot hits the ground at time $t = t_{\text{step}}$ (equation 5).
- Vertical ground reaction force for the foot on the ground in single stance should be positive. This is enforced at the grid points. This gives $N + 1$ inequality constraints.
- Torques in the hip and ankle motors have to be within the actuator limits These are enforced at grid points and correspond to $2(N + 1)$ inequality constraints.
- There are singularities in the governing equations and in the behavior of the optimal solution near $q_2$ or $q_4$ equal zero or $\pi$. To avoid this, the ankle angle was constrained to be $\pi/2$ at heel strike. That is $q_4^{hs} = \pi/2$.

**Methods.** With the initial conditions in single stance $\chi_{ss}^{i}(t = 0)$, we integrate equations of motion for single stance till we reach the end of single stance at time $t = t_{\text{step}}$. After we reach the end of single stance, we apply the heel-strike condition to get the state after heel-strike $\chi_{hs}^{+}(t = t_{\text{step}})$. The optimization software varies the various optimization parameters listed above, in an attempt to minimize the cost given in equation 48, while respecting the various constraints described earlier. Again, we used the sequential quadratic programming package SNOPT [6] which we use later for our controlled-robot optimization.

### 10.2.2 Results

We present results obtained for the step velocity of $V = 0.4$ and step length of $D = 0.4$. Srinivasan [10] pp. 24-25 gives formulas for calculating the COT at various step length and step velocity combinations. Using his formulas, we calculated the ideal point-mass COT to be 0.009882649139799 at $V = D = 0.4$.

Using Ranger's reduced point-mass model and using a grid size $N = 12$, we calculated the COT to be 0.010113205021986. The error between our result and analytical calculations is about 2%. This error is consistent with numerical optimization results presented by Srinivasan [10] on pp. 66 Table 3.1, also taking account that this model, as opposed to the point-mass comparison, has a small, but non-zero leg swing cost.

Figure 34 shows the trajectories for the ankle joints position and velocity, the actuator torques and the mechanical power versus time. The hip motion is low power throughout, due to the light legs. Almost all the energy for walking goes to the ankles to generate the push-off. By increasing the grid size $N$, we found that that the push-off becomes more pre-emptive, decreasing its duration and increasing the peak. This results suggests that the push-off tends to an impulse as the grid is made to grow infinitely big. These results are in agreement with those found by Srinivasan and Ruina [11].

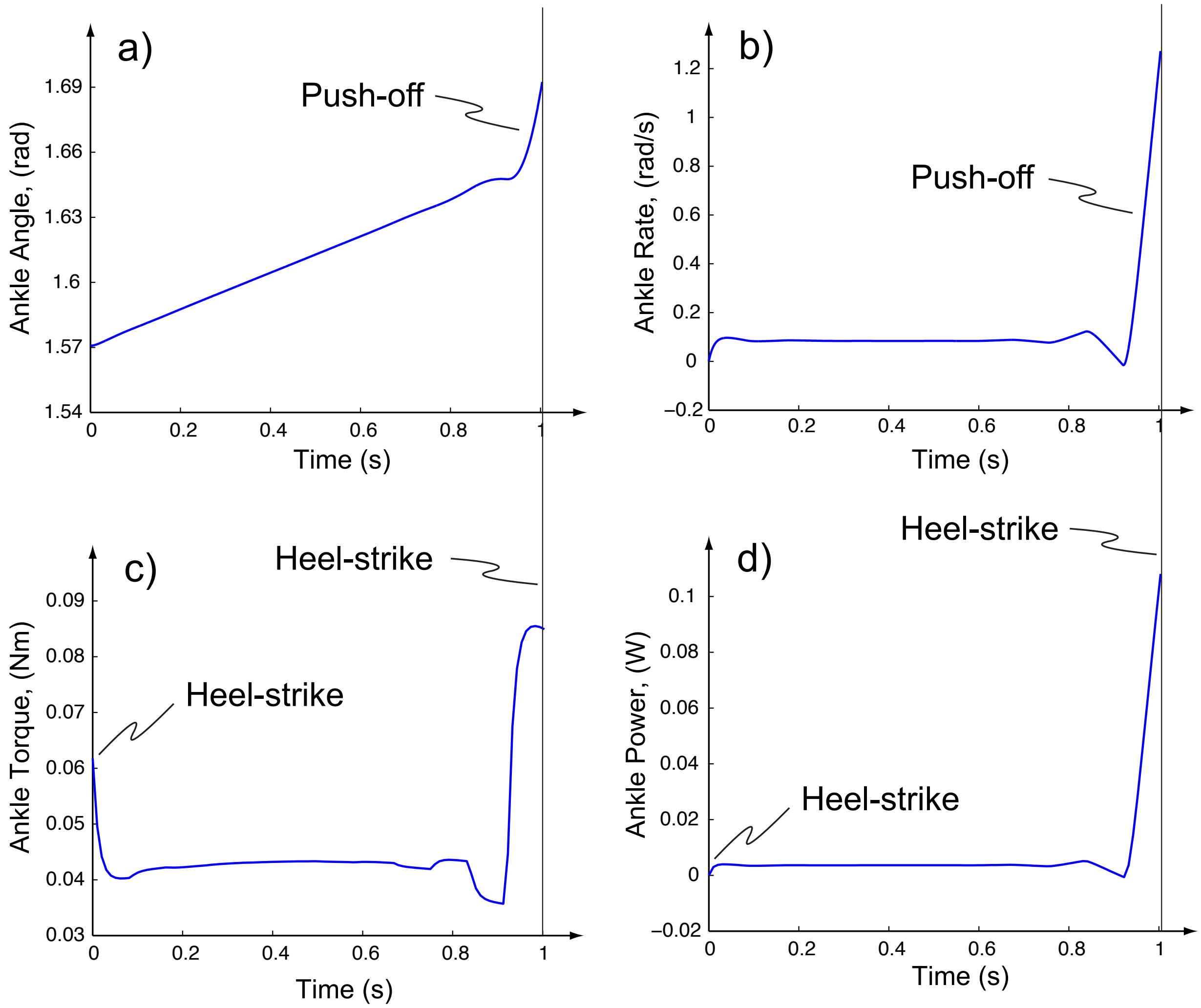


Figure 34: **Ankle trajectory and controls for point-mass model limit of the Ranger model.** The results of the optimization of the Ranger model, with parameters chosen to mimic a point-mass model, are shown. **a)** The ankle angle shows sudden lengthening at push-off; **b)**The ankle rate, being near constant for the small-angle inverted-pendulum phase; **c)** The ankle torque, which has no cost in this model when the ankle rate is zero, the optimizations seeming attempt to discover an impulse is shown by the spike at the right; and **d)** The ankle power, which is effectively zero but for a sudden, seemingly-attempting-to-be-singular rise at push off.

# 11 Trajectory optimization of robot: fine-grid

We use the same optimization software and same general approach described in the previous section to optimize energy use based on the full robot and motor models. As described in the main paper we first do a fine-grid optimization, described in this section, and then a coarse-grid optimization (next section) for use on the robot.

The trajectory optimization for the full robot is infinite dimensional. We resort to a discrete approximation of the problem but with a grid that is fine enough so that, within our model and measurement accuracy, we have achieved an optimum.

## 11.1 Foot-flip energy optimization

**Cost.** We are interested in minimizing the energy for foot-flip and given by,

$$\mathrm{E}_{\text{foot-flip}} = 2\int_{t=t_0=0}^{t=t_f=0.25s} [P_{\text{foot-flip}}]^+ \, dt \tag{49}$$

$P_{\text{foot-flip}}$ is the power in the motor that turns the foot attached to the swing leg in single stance. The factor 2 accounts for flip up and flip down. We assume that our batteries cannot regenerate and to accommodate this fact we define the function $[P]^+$ as follows: $[P]^+ = P$, if $P \geq 0$ and $[P]^+ = 0$, if $P < 0$. The function $[\mathrm{P}]^+$ is non smooth at $P = 0$. We smooth the function using equation 41 as shown in section 9. Also, in the equation above we have heuristically assigned the flip up time to be 0.25 s. This implies that foot flip up and flip down takes a total time of 0.5 s, with additional swing time in the flip up configuration. Consequently, the single stance time has to be at least 0.5 s to prevent foot scuffing.

**Parameters.** The optimization parameters are the motor current as a function of time. We parameterize the current as piecewise linear functions of time. We divide the time into $N^f$ intervals i.e. $t_0 = 0, t_1, \ldots, t_N^f = t_f$, where $t_{i+1} - t_i = 1/N^f$ and $i = 0, 1, ..., N^f$. Thus we have $(N^f + 1)$ unknowns for the current, $I_4^f(t = t_0), I_4^f(t = t_1), \ldots, I_4^f(t = t_{N^f} = t_f)$.

**Constraints.** The optimization constraints are as follows:

$$q_{4m}(t = 0) = 0 \tag{50}$$

$$q_{4m}(t = t_f) = 1.7 \text{ rad} \tag{51}$$

$$\dot{q}_{4m}(t = 0) = 0 \tag{52}$$

$$\dot{q}_{4m}(t = t_f) = 0 \tag{53}$$

$$-8 \text{ A} \leq I_4^f(t_i) \leq 8 \text{ A} \quad \text{where } i = 0, 1, ..., N^f \tag{54}$$

where $q_{4m}$ and $\dot{q}_{4m}$ are the motor position and motor velocity respectively. The 1.7 rad in equation 51 is the angle by which we want the swinging leg's feet to turn in order to clear the worst case obstacle. The actuator current limit of 8 A is specified in constraint equation 54.

**Methods.** With the initial conditions for the motor angle and motor velocity given in 50 and 52, we integrate the equation of motion (see equation 13) of the motor attached to the swinging leg's foot. The output motor torque for use in the equation of motion is calculated using the piecewise linear current profile. The optimization software varies the current parameters at the grid points, while respecting the current bounds in equation 54. The goal of the optimization is to meet the terminal time constraints given in equations 51 and equation 53 and also minimizing the cost $\mathrm{E}_{\text{foot-flip}}$. Again we used SNOPT [6].

## 11.2 Walk COT optimization

**Cost.** We are interested in minimizing the Total Cost of Transport, one term of which is given by

$$\begin{aligned}\mathrm{COT}_{\text{walk}} &= \frac{\mathrm{E}_{\text{walk}}}{M_{\text{tot}}\; g\; d_{\text{step}}} \\ &= \frac{\displaystyle\int_{t=0}^{t=t_{\text{step}}} \sum \left[\mathrm{P}_{\text{walk}}\right]^{+} \mathrm{dt}}{M_{\text{tot}}\; g\; d_{\text{step}}} \end{aligned} \tag{55}$$

as explained in the main paper. $P_{\text{walk}}$ is the power used by the motors but excluding the swing leg's ankle motor in single stance (whose power has already been accounted for in the foot-flip optimization in section 11.1). We assume that our electronics and batteries cannot regenerate; to accommodate this we define the function $[P]^{+}$ as follows: $[P]^{+} = P$, if $P \geq 0$ and $[P]^{+} = 0$, if $P < 0$. The function $[\mathrm{P}]^{+}$ is non smooth at $P = 0$. We smooth the function using equation 41 as shown in section 9.

**Parameters.** The optimization parameters are as follows:

- State at beginning of single stance ($\chi^{i}_{ss}(t=0)$). (see figure 35 (a))
  $\chi^{i}_{ss}(t=0) = [q_1\ ,\ q_2\ ,\ q_{2m}\ ,\ q_3,\ \dot{q}_1\ ,\ \dot{q}_2\ ,\ \dot{q}_{2m}\ ,\ \dot{q}_3]_{ss}(t=0)$. We have skipped the variables $q_4, q_{4m}, \dot{q}_4, \dot{q}_{4m}$ as the swinging leg's foot is decoupled from the rest of the robot in single stance.

- State at the instant before heel-strike ($\chi^{-}_{hs}(t=\alpha t_{\text{step}})$). (see figure 35 (a))
  $\chi^{-}_{hs}(t=\alpha t_{\text{step}}) = [q_1\ ,\ q_2\ ,\ q_{2m}\ ,\ q_3\ ,\ q_4\ ,\ q_{4m}\ ,\ \dot{q}_1\ ,\ \dot{q}_2\ ,\ \dot{q}_{2m}\ ,\ \dot{q}_3\ ,\ \dot{q}_4\ ,\ \dot{q}_{4m}]^{-}_{hs}(t=\alpha t_{\text{step}})$. As the feet are taken to be massless, we assume that the swinging leg's foot motor angle coincides with the swinging leg's ankle angle just before heel-strike. Thus we have $q_4 = q_{4m}$ and $\dot{q}_4 = \dot{q}_{4m}$. Further, we assume that the motor attached to the swinging leg's foot and swinging leg's ankle are at rest just before heel-strike. Thus $\dot{q}_4 = \dot{q}_{4m} = 0$. The initial conditions before heel-strike simplifies to,
  $\chi^{-}_{hs}(t=\alpha t_{\text{step}}) = [q_1\ ,\ q_2\ ,\ q_{2m}\ ,\ q_3\ ,\ q_4\ ,\ q_4\ ,\ \dot{q}_1\ ,\ \dot{q}_2\ ,\ \dot{q}_{2m}\ ,\ \dot{q}_3\ ,\ 0\ ,\ 0]^{-}_{hs}(t=\alpha t_{\text{step}})$.

- Step time ($t_{\text{step}}$)

- Percentage of time spent in single stance ($\alpha$)

- Current in the motors in double stance ($I^{ds}_2(t)$, $I^{ds}_3(t)$, $I^{ds}_4(t)$). We assume piecewise linear currents. In double stance, we divide time into $N^d$ intervals; $t^d_0, t^d_1, \ldots, t^d_{N^d}$. Here $t^d_{i+1} - t^d_i = 1/N^d$, $i = 0, 1, ..., N^d$, $t^d_0 = \alpha t_{\text{step}}$ and $t^d_{N^d} = t_{\text{step}}$. This means we have $3(N^d+1)$ unknowns for currents in double stance; $I^d_j(t=t_0), I^d_j(t=t_1), \ldots, I^d_j(t=t^d_N)$, where $j = 2, 3, 4$.

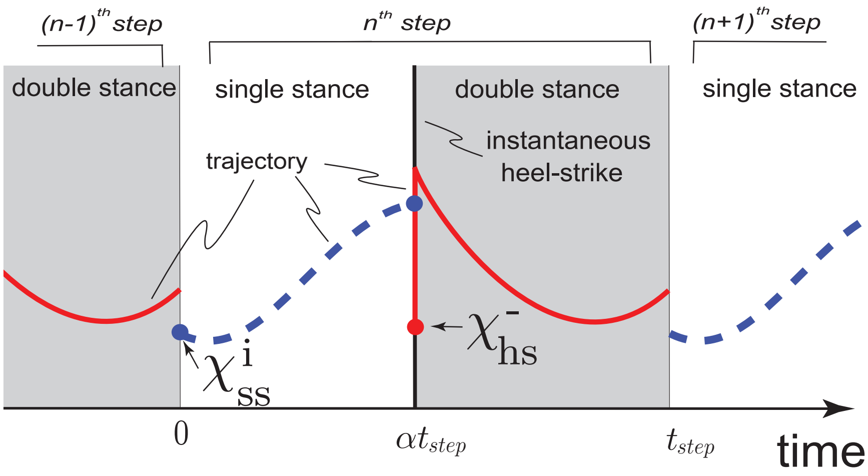
- Current in the motors in single stance ($I_2^{ss}(t)$, $I_3^{ss}(t)$). We assume piecewise linear currents. In single stance, we divide time into $N^s$ intervals; $t_0^s, t_1^s, \ldots, t_{N^s}^s$. Here $t_{i+1}^s - t_i^s = 1/N^s$, $i = 0, 1, ..., N^s$, $t_0^s = 0$ and $t_{N^s}^s = \alpha t_{\text{step}}$. This means we have $2(N^s + 1)$ unknowns for currents in single stance; $I_j^s(t = t_0^s), I_j^s(t = t_1^s), \ldots, I_j^s(t = t_{N^s}^s)$, where $j = 2, 3$.

**Constraints.** The optimization constraints are as follows:

- Periodicity constraints; $\chi_{ss}^i(t = 0) = \chi_{ds}^f(t = t_{\text{step}})$. (see figure 35 (b))
- Periodicity constraints; $\chi_{hs}^-(t = \alpha t_{\text{step}}) = \chi_{ss}^f(t = \alpha t_{\text{step}})$. (see figure 35 (b))
- Transition from single stance to double stance takes place when the swinging leg's foot hits the ground at time $t = \alpha t_{\text{step}}$. This constraint is given by $y_h = y_h$ in equation 5.
- Transition from double stance to single stance takes place when the vertical reaction force on trailing foot goes to zero at time $t = t_{\text{step}}$.
- The current at the motors is continuous in the transition from double stance to single stance; $(I_j^d(t = t_{\text{step}}) = I_j^s(t = 0)$ where $j = 2, 3)$.
- Step velocity is specified, $v_{\text{step}} = V$.
- Step length is specified, $d_{\text{step}} = D$.
- To prevent scuffing at the beginning of single stance phase we impose that the absolute velocity of the hip at the beginning of single stance should not be in the walking direction.
- Vertical ground reaction force for the foot on the ground in single stance should be positive and this is enforced at the grid points. This gives $N^s + 1$ inequality constraints.
- Vertical ground reaction force on both the feet in double stance should be positive, except on the trailing leg at the double stance to single stance transition when it should be zero. These constraints are enforced at the grid points. This gives us $2N^d + 1$ inequality constraints.
- Currents in the motors have to be within the actuator limits ($-8\text{A} \leq I_j^i \leq 8\text{A}$, $i = \text{ss}, \text{ds}$; for ds $j = 2, 3, 4$ and for ss $j = 2, 3$). These are enforced at grid points and correspond to $3(N^d + 1) + 2(N^s + 1)$ inequality constraints.

**Methods.** With the initial conditions in single stance $\chi_{ss}^i(t = 0)$, we integrate equations of motion for single stance till we reach the end of single stance at time $t = \alpha t_{\text{step}}$. The torques for the single stance equations are obtained using the assumed piecewise linear currents in single stance. Similarly, we take the initial conditions just before heel-strike $\chi_{hs}^-(t = \alpha t_{\text{step}})$ and first apply the heel-strike condition to get the initial condition at the beginning of double stance. With the initial conditions in double stance, we integrate equations of motion for double stance till we reach the end of double stance at time $t = t_{\text{step}}$. The torques for use in the equations of motion are obtained using the assumed piecewise linear currents in double stance. The optimization software varies the various optimization parameters listed above, in an attempt to minimize the cost 55, while respecting the various constraints described earlier. We used sequential quadratic programming package SNOPT [6], to solve the optimization problem. The results of this optimization are shown and described in the main paper.

a) State space parameters are $\chi_{\mathrm{ss}}^{\mathrm{i}}$ and $\chi_{\mathrm{hs}}^{-}$


(n-1)th step
nth step
(n+1)th step
double stance
single stance
double stance
single stance
instantaneous heel-strike
trajectory
$\chi_{\mathrm{hs}}^{-}$
$\chi_{\mathrm{ss}}^{\mathrm{i}}$
0
$\alpha t_{step}$
$t_{step}$
time

b) Multiple shooting to enforce periodicity

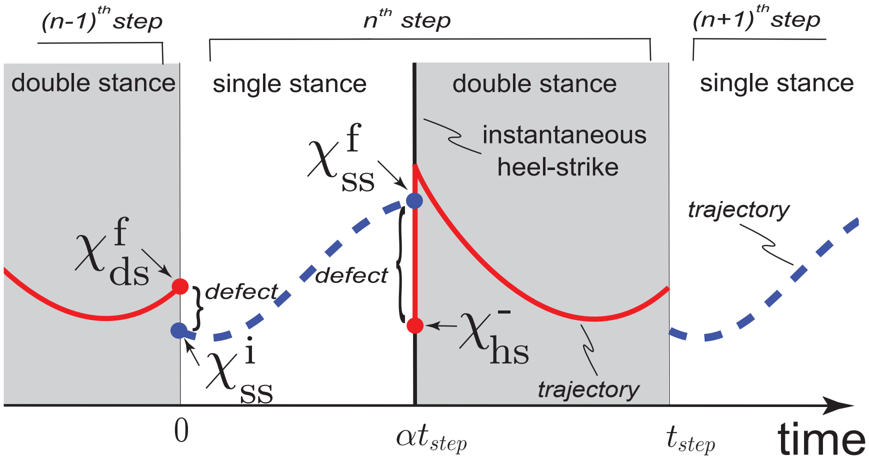

(n-1)th step
nth step
(n+1)th step
double stance
single stance
double stance
single stance
instantaneous heel-strike
$\chi_{\mathrm{ss}}^{\mathrm{f}}$
$\chi_{\mathrm{ds}}^{\mathrm{f}}$
defect
defect
trajectory
$\chi_{\mathrm{hs}}^{-}$
$\chi_{\mathrm{ss}}^{\mathrm{i}}$
trajectory
0
$\alpha t_{step}$
$t_{step}$
time

Figure 35: **Multiple shooting method to solve optimal trajectory control problem.** (a) Initial condition at the beginning of single stance phase $\chi_{ss}^{i}$ and just before heelstrike $\chi_{hs}^{-}$ are the state space optimization parameters. The step time is $t_{\text{step}}$ and fraction of time spent in single stance is $\alpha$. (b) To enforce periodicity we equate the state in the beginning of single stance to that at the end of the double stance, i.e. $\chi_{ss}^{i}(t=0) = \chi_{ds}^{f}(t=t_{\text{step}})$ and the state at the instant before heel-strike to that at the end of single stance $\chi_{hs}^{-}(t=\alpha t_{\text{step}}) = \chi_{ss}^{f}(t=\alpha t_{\text{step}})$. That is the optimization drives the defects, shown as discontinuities between red (solid) and blue (dashed) curves in the figure, to zero.

# 12 Trajectory optimization: coarse-grid

To simplify the control implemented by the main brain (ARM 9), the motor current trajectories are defined with far few parameters than are used in the fine-grid optimization. Rather than try to approximate the fine-grid trajectories with the coarse-grid trajectories, a new optimization is done with the coarse-grid description. This is described in section 6 of the main paper with some more details provided here.

**Problem formulation.** We simplify the problem by separating the foot flip-up and flip-down (during swing) optimization from the optimization of other costs. This is a reasonable approximation because the feet have such low mass that their motions during swing are negligibly coupled to the other motion of the robot.

We carry out two sequential optimizations as follows.

1. We approximate the current profile needed to do the foot-flip optimization. This optimization gives us the energy needed to do the foot-flip using the approximate control representation ($\mathrm{E}^{\text{const}}_{\text{coarse-grid-foot-flip}}$).
2. Using the energy for flip-foot optimization ($\mathrm{E}^{\text{const}}_{\text{coarse-grid-foot-flip}}$) and using an approximation to the converged current profile for the walk optimization presented in 11.2 we minimize the total cost of transport (TCOT).

We elaborate on the details of these optimization in the next two sections.

## 12.1 Foot-flip energy optimization

**Cost, constraints and methods.** These are the same as those presented in 11.1. The result of this optimization will give the energy needed for doing the foot-flip using the approximate representation $\mathrm{E}^{\text{const}}_{\text{coarse-grid-foot-flip}}$.

**Parameters.** We want the foot on the swinging leg to get to a prescribed position for foot clearance. Hence we chose to do a proportional derivative control on motor position for the foot-flip. Thus, we have three parameters; proportional gain, derivative gain, and reference angle.

## 12.2 Cost of transport minimization

**Cost.** We are interested in minimizing the total cost of transport (COT) and defined as,

$$\begin{aligned}\mathrm{TCOT} &= \frac{\text{Energy used per step}}{\text{Weight} \times \text{Step length}}\\ &= \frac{\mathrm{P}_{\text{fixed}}\mathrm{t}_{\text{step}} + \mathrm{E}^{\text{const}}_{\text{coarse-grid-foot-flip}} + \mathrm{E}_{\text{walk}}}{M_{\text{tot}}\; g\; d_{\text{step}}}\\ &= \frac{\mathrm{P}_{\text{fixed}}\mathrm{t}_{\text{step}} + \mathrm{E}^{\text{const}}_{\text{coarse-grid-foot-flip}} + \displaystyle\int_{t=0}^{t=t_{\text{step}}} \sum\, [\mathrm{P}_{\text{walk}}]^{+}\, \mathrm{dt}}{M_{\text{tot}}\; g\; d_{\text{step}}}\end{aligned}$$

Here $\mathrm{P}_{\text{fixed}}$ is the power need to run the sensors, microprocessors, and motor controllers and is fixed at 5.15 W. $\mathrm{E}^{\text{const}}_{\text{coarse-grid-foot-flip}}$ is the energy needed to flip the foot on the swing leg to do ground clearance, is a constant per step, and is obtained from foot-flip optimization presented in 12.1. $\mathrm{E}_{\text{walk}}$ is the energy needed to take a step. This is calculated by taking time integral of the sum of power of motors involved in taking a step.

**Parameters.** The optimization parameters are as follows:

- State at beginning of single stance ($\chi^i_{ss}(t=0)$). See figure 35 (a)).
  $\chi^i_{ss}(t=0) \;=\; [q_1\;,\;q_2\;,\;q_{2m}\;,\;q_3,\;\dot{q}_1\;,\;\dot{q}_2\;,\;\dot{q}_{2m}\;,\;\dot{q}_3]_{ss}(t=0)$
- State at the instant before heel-strike ($\chi^-_{hs}(t=\alpha t_{\text{step}})$). See figure 35 (a).
  $\chi^-_{hs}(t=\alpha t_{\text{step}}) \;=\; [q_1\;,\;q_2\;,\;q_{2m}\;,\;q_3\;,\;q_4\;,\;q_4\;,\;\dot{q}_1\;,\;\dot{q}_2\;,\;\dot{q}_{2m}\;,\;\dot{q}_3\;,\;0\;,\;0]^-_{hs}(t=\alpha t_{\text{step}})$
- Step time ($t_{\text{step}}$)
- Percentage of time spent in different phases; $\alpha$, $\beta$, $\gamma$ such that $0 < \alpha, \beta, \gamma < 1$. See figure 36.
- We divide the walking cycle into a finite state machine as shown in figure 36; one for the hip and one for each foot. Each state machine is divided into a state and given a name (e.g. pre-mid swing, push-off). Each state has a suitable current parameterization and is shown in the parenthesis. Transitions for states are shown along the time axis.

  **Hip state machine (top panel):** is divided into three states; a pre-mid swing state, a post-mid state, and a double stance state. In the pre-mid swing state we apply a constant current, while the post-mid swing and double stance states are both passive. We start with pre-mid swing state. When the robot's stance leg is vertical (in the single stance phase), the pre-mid swing to post-mid swing transition occurs. On heel-strike, we get the post-mid swing to double stance state transition. Finally, the double stance to pre-mid swing transition leading to the next step occurs when the vertical reaction force on the trailing leg goes to zero (also called the toe-off).

  **Foot state machine (middle panel):** is the state machine for the foot which stays on the ground for the full step. This foot has three states; single stance, push-off, and double stance. For single stance we do a proportional-derivative (PD) control on the absolute foot angle. For the push-off we do a proportional (P) control on the absolute foot angle. The double stance state is passive. We start with the state single stance. When the swinging leg's ankle reaches a critical height the single stance to push-off transition occurs. On heel-strike, the push-off to double stance transition occurs. Finally, when the vertical reaction force on the trailing leg goes to zero i.e. on toe-off, we get the double stance to flip-up transition leading to the next step.

  **Foot state machine (bottom panel):** is the state machine for the foot that is in the air during single stance and is on the ground during double stance. This foot has three states; flip-up, flip-down, and double stance. For double stance we do a P-D control on absolute foot angle. In the flip-up and flip-down we do a P-D control on the foot angle. Since the foot is massless, we decoupled the motion of the swinging leg's foot from the rest of the robot simulation. We did a separate optimization for the flip-up and flip-down states and is presented in section 12.1. We start with flip up position for this foot at time t=0. We transition from flip-up to flip-down state when the hip reaches a certain critical angle. On heel-strike, we transition from flip down to double stance

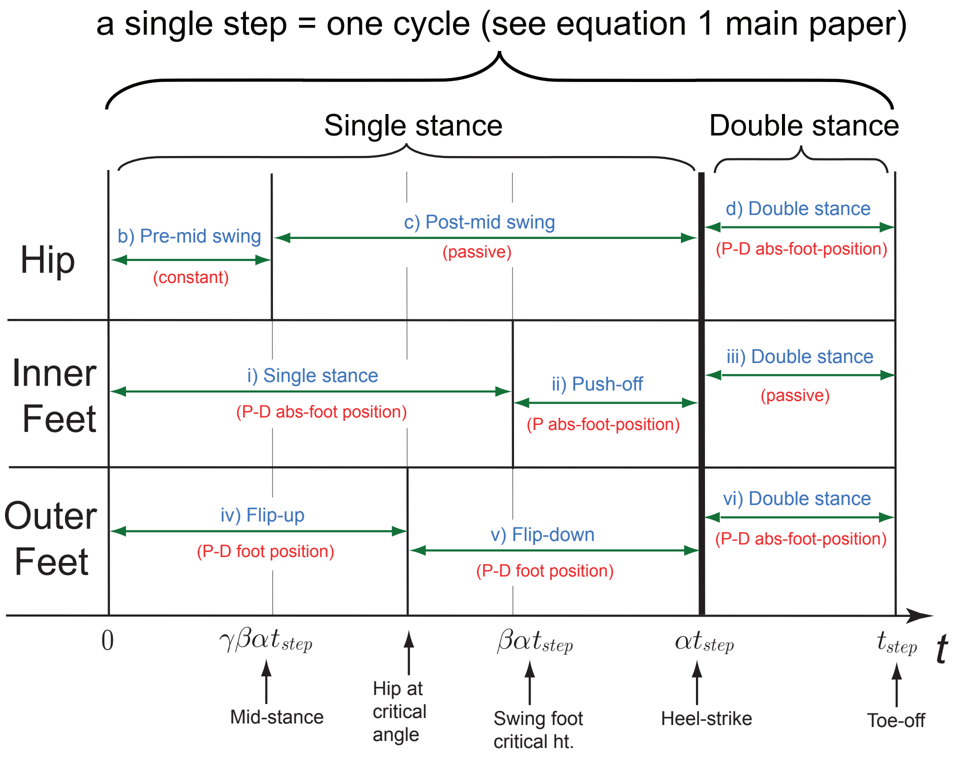

a single step = one cycle (see equation 1 main paper)
Single stance
Double stance
Hip
b) Pre-mid swing
(constant)
c) Post-mid swing
(passive)
d) Double stance
(P-D abs-foot-position)
Inner Feet
i) Single stance
(P-D abs-foot position)
ii) Push-off
(P abs-foot-position)
iii) Double stance
(passive)
Outer Feet
iv) Flip-up
(P-D foot position)
v) Flip-down
(P-D foot position)
vi) Double stance
(P-D abs-foot-position)
0
$\gamma\beta\alpha t_{step}$
$\beta\alpha t_{step}$
$\alpha t_{step}$
$t_{step}$
$t$
Mid-stance
Hip at critical angle
Swing foot critical ht.
Heel-strike
Toe-off

Figure 36: **Finite state machine used to do the coarse-grid optimization.** We show the state machine for only one step. A step is divided into single stance and double stance using the parameter $\alpha$ ($0 < \alpha < 1$ ). $\alpha$ denotes the fraction of step time spent in single stance. Further, the single stance phase is divided into three phases using the parameters $\beta$ and $\gamma$ ($0 < \beta, \gamma < 1$). Each state is given a name e.g. pre-mid swing, push-off. The control action associated with the state is shown in parenthesis. State transitions conditions are written on the time axis.

state. The double stance to single stance transition occurs when the vertical reaction force on trailing legs goes to zero (also called toe-off) leading to the next step.

**Constraints.** The constraints for this optimization are same as those presented for the optimal control problem presented in section 11.2 but with a few exceptions. We do not have the step length and step velocity constraint. Instead, we bound the hip angle at heel-strike to have a maximum value of 0.4 rad (to prevent robot wear and tear associated with big step lengths). We do not have a grid to enforce the vertical ground reaction force and current constraints. So, we create a grid of size $N^{s1}$ for time interval $0 \leq t \leq \gamma\beta\alpha t_{\text{step}}$, $N^{s2}$ for time interval $\gamma\beta\alpha t_{\text{step}} \leq t \leq \beta\alpha t_{\text{step}}$, and $N^{s3}$ for time interval $\beta\alpha t_{\text{step}} \leq t \leq \alpha t_{\text{step}}$, all these are in single stance phase. We also create a grid of size $N^{d}$ for the time interval $\alpha t_{\text{step}} \leq t \leq t_{\text{step}}$ in double stance phase. Now we enforce the reaction force and current constraints on this grid in a similar way as that presented in section 11.2.

**Methods.** Same as those presented in section 11.2.

# 13 Finite state machine for high level walk control

We use a concurrent, hierarchical finite state machine to code our combined coarse-grid and reflex based discrete controller on the robot.

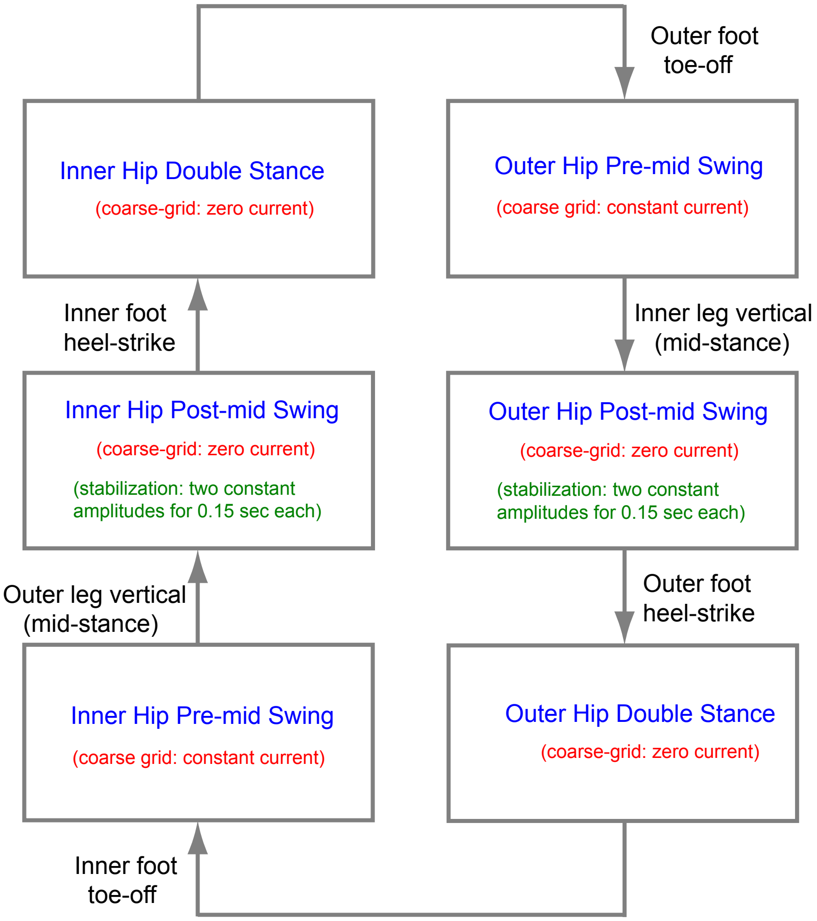

Outer foot toe-off
Inner Hip Double Stance
(coarse-grid: zero current)
Outer Hip Pre-mid Swing
(coarse grid: constant current)
Inner foot heel-strike
Inner leg vertical (mid-stance)
Inner Hip Post-mid Swing
(coarse-grid: zero current)
(stabilization: two constant amplitudes for 0.15 sec each)
Outer Hip Post-mid Swing
(coarse-grid: zero current)
(stabilization: two constant amplitudes for 0.15 sec each)
Outer leg vertical (mid-stance)
Outer foot heel-strike
Inner Hip Pre-mid Swing
(coarse grid: constant current)
Outer Hip Double Stance
(coarse-grid: zero current)
Inner foot toe-off

Figure 37: **Hip finite state machine.**

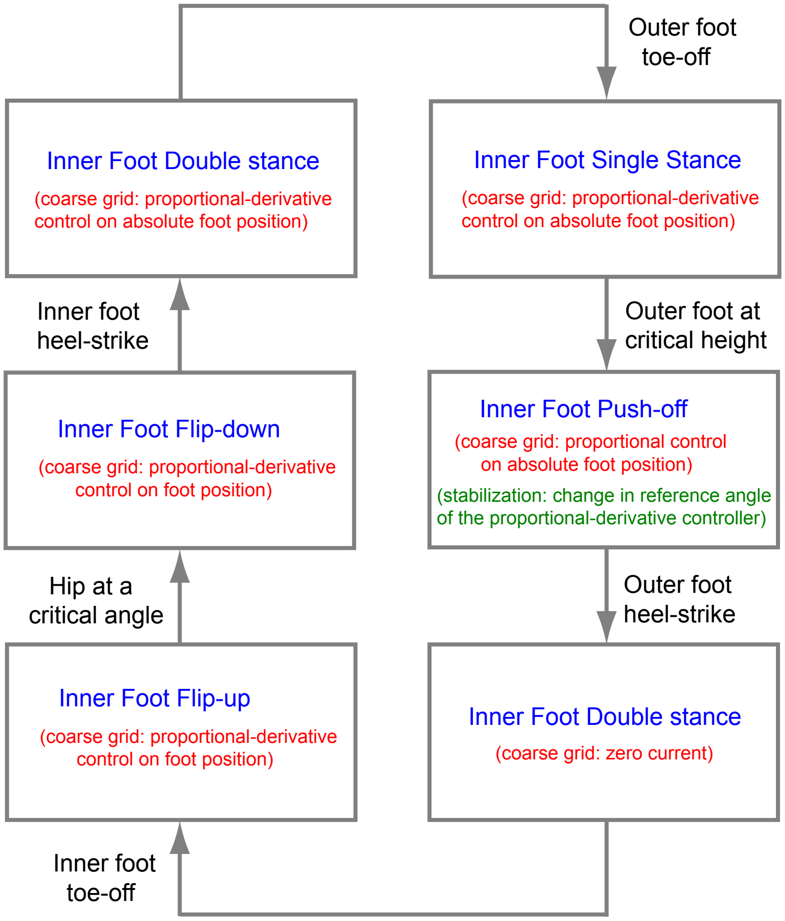

Outer foot
toe-off
Inner Foot Double stance
(coarse grid: proportional-derivative control on absolute foot position)
Inner Foot Single Stance
(coarse grid: proportional-derivative control on absolute foot position)
Inner foot
heel-strike
Outer foot at
critical height
Inner Foot Flip-down
(coarse grid: proportional-derivative control on foot position)
Inner Foot Push-off
(coarse grid: proportional control on absolute foot position)
(stabilization: change in reference angle of the proportional-derivative controller)
Hip at a
critical angle
Outer foot
heel-strike
Inner Foot Flip-up
(coarse grid: proportional-derivative control on foot position)
Inner Foot Double stance
(coarse grid: zero current)
Inner foot
toe-off

Figure 38: **Inner foot finite state machine.**

# References

[1] M.J. Coleman. *A stability study of a three-dimensional passive-dynamic model of human gait.* PhD thesis, Cornell University, 1998.

[2] M.J. Coleman. Numerical accuracy case studies of two systems with intermittent dynamics a 2d rimless spoked wheel and a 3d passive dynamic model of walking. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Application and Algorithms*, 16:59–87, 2009.

[3] C. Duan and R. Singh. Dynamics of a 3dof torsional system with a dry friction controlled path. *Journal of sound and vibration*, 289(4-5):657–688, 2006.

[4] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman. The simplest walking model: Stability, complexity, and scaling. *ASME J. of Biomech. Eng.*, 120:281–288, 1998.

[5] M.S. Garcia. *stability, scaling, and chaos in passive-dynamic gait models.* PhD thesis, Cornell University, 1999.

[6] P.E. Gill, W. Murray, and M.A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[7] T. McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

[8] T. McGeer. Passive dynamic biped catalogue. In *In Proc. of 2nd International Symposium on Experimental Robotics*, pages 465–490, 1991.

[9] A. Ruina et al. Cornell ranger 2011, 4-legged bipedal robot. [available online]. `http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/ranger/Ranger2011/`. Or Google search for: cornell ranger, April 2012.

[10] M. Srinivasan. *Why walk and run: energetic costs and energetic optimality in simple mechanics-based models of a bipedal animal.* PhD thesis, Cornell University, 2006.

[11] M. Srinivasan and A. Ruina. Computer optimization of a minimal biped model discovers walking and running. *Nature*, 439(7072):72–75, 2005.

[12] K. Taji and M. Miyamoto. A globally convergent smoothing newton method for nonsmooth equations and its application to complementarity problems. *Computational Optimization and Applications*, 22:81–101, 2002.